

An Address-Based Routing Scheme for Static Applications of Wireless Sensor Networks

Weibo Li

A thesis submitted in partial fulfilment
of the requirements for the degree of
Master of Engineering
in
Electrical and Computer Engineering
at the
University of Canterbury,
Christchurch, New Zealand.

April 2008

ABSTRACT

Wireless sensor networks (WSNs), being a relatively new technology, largely employ protocols designed for other ad hoc networks, especially mobile ad hoc networks (MANETs). However, on the basis of applications, there are many differences between WSNs and other types of ad hoc network and so WSNs would benefit from protocols which take into account their specific properties, especially in routing. Bhatti and Yue (2006) proposed an addressing scheme for multi-hop networks. It provides a systematic address structure for WSNs and allows network topology to avoid the fatal node failure problem which could occur with the *ZigBee* tree structure. In this work, a new routing strategy is developed based on Bhatti and Yue's addressing scheme. The new approach is to implement a hybrid flooding scheme that combines flooding with shortest-path methods to yield a more practical routing protocol for static WSN applications. The primary idea is to set a flooding counter K as an overhead parameter of control messages which are used to discover routes between any arbitrary nodes. These route request messages are flooded for K hops and then oriented by shortest-path routing from multiple nodes in the edge of the flooding area to the destination. The simulation results show that this protocol under certain wireless circumstances is more energy conscious and produces less redundancy than reactive ZigBee routing protocol. Another advantage is that the routing protocol can adapt any dynamic environment in various WSN applications to achieve a satisfactory data delivery ratio in exchange for redundancy.

Keywords - Wireless Sensor Networks; ZigBee; Ad hoc networks; Routing Protocol; Static Applications; Adaptive Flooding

ACKNOWLEDGEMENT

Intelligent people are everywhere, but a wise man is hard to meet. I am lucky to know two wise men at one time, Professor Harsha Sirisena and Professor Krys Pawlikowski. I would like to start by thanking Professor Harsha Sirisena, my supervisor and mentor. Harsha has been a wonderful mentor in the past two years. I prefer to call him a "Guru" who not only teaches me sciences but also influences me with wisdom. He is such a kind person who really understands and cares the students from overseas like me. I thank him for providing me many opportunities, support, encouragement and guidance. What I learnt from him will be of great benefit to the rest of my life. I would also like to thank Professor Krys Pawlikowski, my joint supervisor and good friend. Krys is always energetic and optimistic, which makes him a good mentor for every young man. I thank him for his energy and time invested into my research and our Networking Research Group. Krys is also my good friend. My first river rafting experience was shared with him, which was a wonderful time.

I would also like to thank my colleagues. I can always get help and new ideas from them. It was really a good time to work with them for two years.

I am indebted to my parents, Li Shigui and Shi yanhua, for everything I own now. They gave me a heart and a soul, taught me the meaning of life, the importance of family. I would like to thank them for their understanding and support, they gave me their wings and made me fly in pursuit of my dreams. They did what the best parents in the world could do.

Finally, I would like to thank my dear wife, Su ying. She is the only reason that I did

all of this. She dedicates herself to me and the family and supports me both materially and mentally. She always bears hardship by herself without complaint and gives all the best to me. I believe there is nothing I can not achieve with her by my side.

CONTENTS

ABSTRACT	i
ACKNOWLEDGEMENT	iii
CHAPTER 1 INTRODUCTION	1
1.1 Wireless Sensor Networks and ZigBee	1
1.2 Wireless Sensor Network Applications	2
1.3 General Design Principles and Challenges	7
1.4 Wireless Sensor Network Routing Protocols	9
1.4.1 Taxonomy of Ad Hoc Routing Protocols	10
1.4.2 Taxonomy of WSN Routing Protocols	11
1.5 Addressing Schemes	13
1.6 Dissertation Overview	15
CHAPTER 2 PRIOR AND RELATED WORKS	17
2.1 ZigBee Routing Discovery and Maintenance Algorithms	17
2.1.1 Route discovery	17
Initiation of Route Discovery	18
On Receipt of a Route Request	19
On Receipt of a Route Reply	23
2.1.2 Route Maintenance	23
2.2 A Structured Addressing Scheme	25
2.2.1 Scheme Description	26
2.2.2 Discussion	30
CHAPTER 3 ADDRESS-BASED ADAPTIVE FLOODING (<i>ABAF</i>) ROUTING PROTOCOL	33
3.1 Motivation and Problem Definition	33
3.2 Scheme Scenario	35
3.2.1 Network Model	35
3.2.2 Network Configuration	38
3.3 Scheme Description	40
3.3.1 Shortest Path Phase	40
3.3.2 Partial Flooding	42
3.3.3 Hybrid Adaptive Flooding	45

3.3.4	The Route Discovery Operation of <i>ABAF</i>	51
	Initiation of Route Discovery	52
	On Receipt of a Route Request	53
3.3.5	Tentative Scheme on Route Maintenance	54
CHAPTER 4	SIMULATION AND RESULTS ANALYSIS OF <i>ABAF</i>	59
4.1	Simulator	59
4.2	Ad hoc Sim	62
4.2.1	Mobility Modules	63
4.2.2	PHY Layer	65
4.2.3	MAC Layer	66
4.2.4	AODV Routing Module	67
4.2.5	APP Layer	69
4.3	Simulating AODV in A Static Network	70
4.3.1	Fixed Topology	70
4.3.2	Transmitter Module	73
4.3.3	Modifications on AODV Routing Module	76
4.4	<i>ABAF</i> Model	76
4.5	Simulation Analysis and Performance Evaluation	79
4.5.1	Simulation Analysis and Setup	79
4.5.2	Results Evaluation	82
	Delivery Ratio	83
	Redundancy	85
CHAPTER 5	CONCLUSIONS	89
5.1	Summary of Contributions	89
5.2	Future Works	90
REFERENCES		93

LIST OF FIGURES

1.1	ZigBee Network Topologies	2
1.2	ZigBee Protocol Stack Architecture	3
1.3	IEEE 802 Family	3
1.4	Vineyard Monitoring and Irrigation System	5
1.5	ZigBee Applications	6
1.6	Sensor Web 1.0 Pod	9
1.7	Hierarchical and Flat Address Spaces	14
2.1	Basic ZigBee Route Discovery Algorithm	20
2.2	Receipt of Route Request	22
2.3	Receipt of Route Reply	24
2.4	3-dimensional Hypercube Structure	26
2.5	Addressing Process in A 2-dimensional Address Grid	28
2.6	An Example of A Chain Address Structure	29
2.7	Disordered Address Assignment	31
3.1	A single node joining in WSNs	34

3.2	Implementation of Bhatti and Yue's Addressing Scheme	37
3.3	The <i>ABAF</i> Network Model	38
3.4	A network section utilized for analysis	39
3.5	Shortest-path Routing	41
3.6	Partial Flooding	43
3.7	Guaranteed Message Delivery	44
3.8	The Route Discovery Process of The <i>ABAF</i> Routing Protocol in a regular $(N+1) \times (N+1)$ grid	47
3.9	Basic <i>ABAF</i> Route Discovery Algorithm	53
3.10	Receipt of Route Request in <i>ABAF</i>	55
4.1	Simulators for Different Application Domains	60
4.2	The Simulation Topology and The Host Internal Structure of Ad hoc Sim	63
4.3	The Connection between PHY Modules of Two Nodes	65
4.4	The Input Dialog Boxes for The Number of Columns and Rows	71
4.5	A 4×3 Static model Generated by FAS	72
4.6	The Topology with Dynamic Links Generated by The Transmitter Module	75
4.7	Delivery Ratio in A 5×5 Network with A Link Failure Probability of 0.05	83
4.8	Delivery Ratio in A 20×20 Network with A Link Failure Probability of 0.05	84
4.9	Delivery Ratio in A 5×5 Network with A Link Failure Probability of 0.20	84

4.10 Delivery Ratio in A 20×20 Network with A Link Failure Probability of 0.20	85
4.11 Redundancy in A 5×5 Network with A Link Failure Probability of 0.05	86
4.12 Redundancy in A 20×20 Network with A Link Failure Probability of 0.05	86
4.13 Redundancy in A 5×5 Network with A Link Failure Probability of 0.20	87
4.14 Redundancy in A 20×20 Network with A Link Failure Probability of 0.20	87
4.15 Received DATA in A 20×20 Network with A Link Failure Probability of 0.05	88

LIST OF TABLES

2.1	Comparison of Addressing Schemes	32
3.1	Neighbor Table Entry	57
3.2	Routing Table	57
3.3	Routing Table Entry for The Four Valid Links	57
4.1	Algorithm 1: The class <i>setPos</i>	71
4.2	Algorithm 2: The Modified Network Module <i>world.ned</i>	72
4.3	Algorithm 3: The Modified PHY Module	72
4.4	Algorithm 4: The Transmitter Module	75
4.5	Algorithm 5: PHY Handles a Message from Transmitter	76
4.6	Algorithm 6: <i>ABAF</i> algorithm	78
4.7	Algorithm 7: The class <i>getNexthop</i>	79
4.8	The Old Setting of The Key Parameters	80
4.9	Simulation Parameters	81

Chapter 1

INTRODUCTION

1.1 WIRELESS SENSOR NETWORKS AND ZIGBEE

Wireless networks enable us to communicate with each other anytime and anyplace. The wireless networks commonly used are known as infrastructured networks. This type of network always employs central devices known as base stations to exchange data with mobile handsets within the transmission range. When a mobile handset moves out of coverage of a base station, a handoff process occurs to ensure the communication continues via a new base station. The other type of network is infrastructureless, called ad hoc networks. Ad hoc networks have no fixed base station: all nodes connect to each other dynamically. As defined by ZigBee Alliance, each node functions as a router to discover and maintain routes, as shown in Figure 1.1. When network components work in a mobile manner, the network is called Mobile Ad Hoc Network (MANET). MANETs focus on End-to-End communication, while the devices in static networks cooperate to accomplish some tasks. Typical examples of an ad hoc network are Bluetooth and Wireless Sensor Networks [1].

Wireless sensors have been developed for decades, and they are mainly used in static networks. All wireless sensor nodes collaborate to achieve application goals. Therefore, many principle differences exist between a Wireless Sensor Network (WSN) and a Mobile Ad Hoc Network (MANET) [2]. In recent years, the development of the technologies of semiconductors, batteries and microprocessors and the demands of commercial applications have really pushed WSNs to the front line of wireless network research.

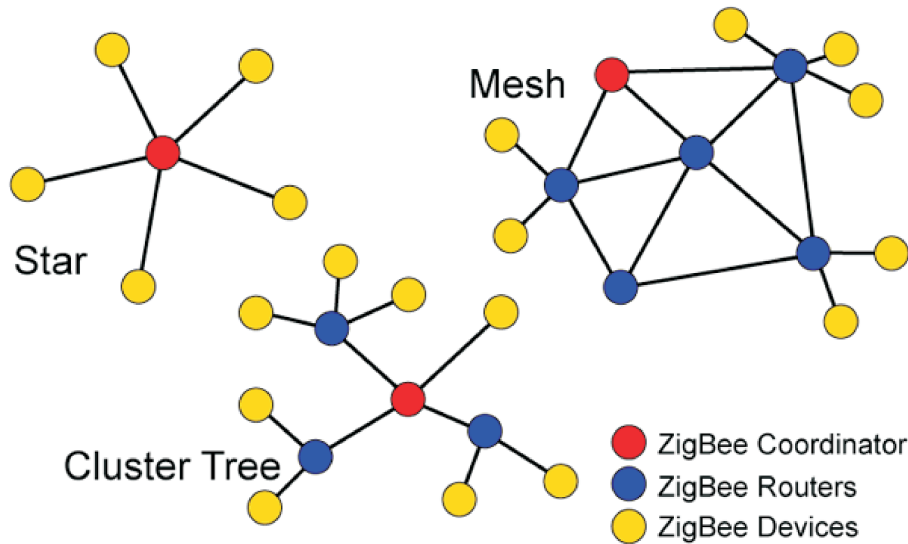


Figure 1.1 ZigBee Network Topologies

ZigBee is a suite of up-level communication protocols based on the IEEE 802.15.4 standard for Wireless Sensor Network, see Figure 1.2 [3]. IEEE 802.15.4 provides the base layers which includes a physical (PHY) layer and a medium access control (MAC) layer, while ZigBee defines protocols from the network (NWK) layer to a part of the application (APP) layer. The top application layer is reserved to end users.

Compared with its brother, the well-known Bluetooth which employs IEEE 802.15.1 standard, ZigBee has the properties of a long transmission range, low bandwidth, low cost, low power consumption, high flexibility and high scalability [1]. The distinctive properties of ZigBee just meet the exact requirements of particular applications and therefore make ZigBee a popular research area. Figure 1.3 illustrates the position of ZigBee in IEEE wireless space [4].

1.2 WIRELESS SENSOR NETWORK APPLICATIONS

A Wireless Sensor Network emphasizes more the concept of a whole network rather than a set of individual nodes, which means all wireless sensor nodes are required to cooperate in order to function. Hogler and Andreas (2005) give an appropriate def-

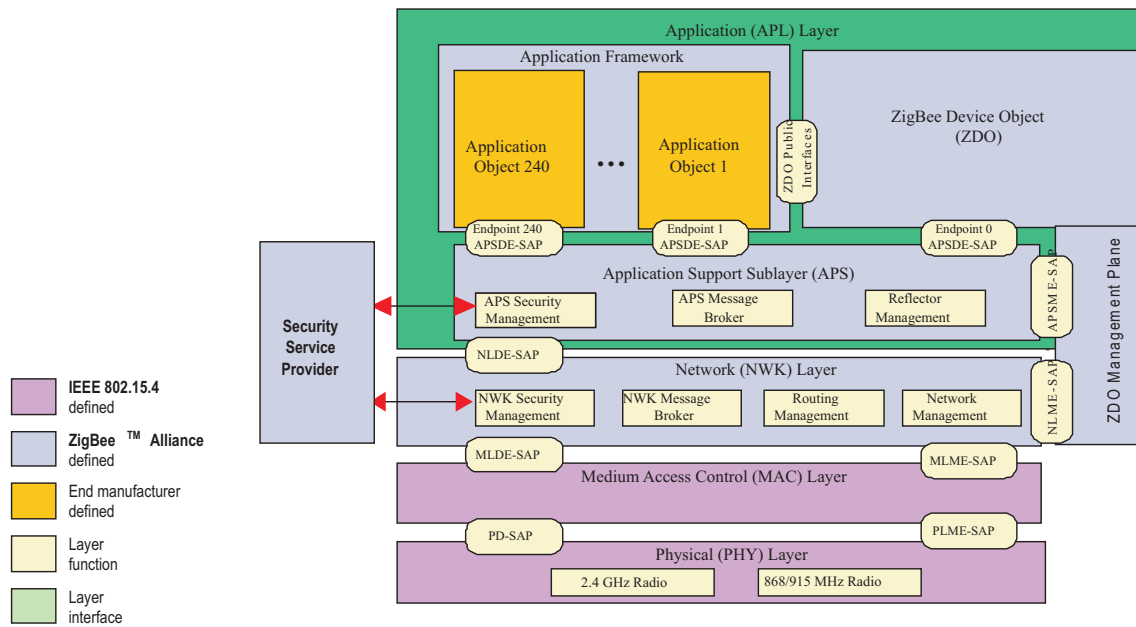


Figure 1.2 ZigBee Protocol Stack Architecture

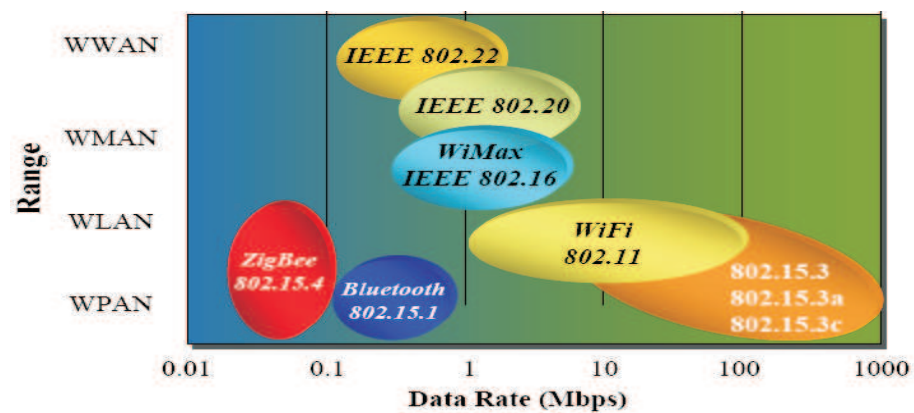


Figure 1.3 IEEE 802 Family

inition of Wireless Sensor Networks, which ”*consist of individual nodes that are able to interact with their environment by sensing or controlling physical parameters; these nodes have to collaborate in order to fulfill their tasks as, usually, a single node is incapable of doing so; and they use wireless communication to enable this collaboration*” (pp.2). The unique characteristics of sensor nodes make WSNs highly flexible; therefore, the range of application is very wide. Due to the immaturity of Wireless Sensor Network standards, many related protocols and technologies are currently used in WSN architecture and produces many possible WSN application scenarios. The following examples are a brief overview of WSN application cases which indicate the scope of WSN applications, which are also categorized in [5] into a *near-term* commercial applications list according to research and application interests in the US.

- **Auto Meter Reading (AMR)** Wireless sensor nodes are embedded in metering devices to record and transmit data periodically. It can be often found in gas or electricity metering within a residential community range. The metering data of the whole area can be read by a handset device, without moving around. The handset device can even be mounted in a station to collect and transmit data automatically through a telephone line or the Internet.
- **Environment and Agriculture Monitoring** Wireless sensor nodes can be distributed in woods, vineyards or marine farms to monitor the environmental conditions and report details. Furthermore, the sensor nodes can act as an actuator connected to irrigating machines to achieve automatic irrigation.
- **Intelligent Buildings** In complex buildings, energy is actually wasted by inefficient usage of air conditioning systems. With the feedback from sensors installed in corners of offices and corridors, air conditioners can self-operate accordingly. It can not only improve the comfort level of an indoor environment but also reduce the power consumption [6]. In addition, connecting computers and remote controllers to electronic equipment and appliances, such as projectors, light switches, printers and TVs has the ability to achieve home and office automation.

- **Industrial Detection and Control** Sensors play an important role in industry. Wireless sensors can be dropped or fixed to areas that are unreachable or hard to access, in order to monitor industrial metrics and detect mechanical faults. Moreover some dangerous operations can be done by remote control through WSNs.
- **Stock Management** In supermarkets or docks, electronic labels can be attached to shelves and containers. Warehouse managers can easily upload and download freight information by a handset or monitor and track freights by a security system based on a WSN.



Figure 1.4 Vineyard Monitoring and Irrigation System. The deployed wireless device is a node of Camalie Net which is a soil moisture monitoring system developed by UC Berkeley in collaboration with Intel Corp and commercialized by Crossbow Inc.

The potential forms of WSN are almost unlimited. Figure 1.4 gives an illustration of WSN application in a vineyard [7]. A good summary of general WSN applications can be found in Bob Heile's tutorial [4], also see Figure 1.5.

Most of these applications share similar characteristics so that they are taxonomized into the following types in [8, 2].

- **Event Monitoring and Detection** Sensor nodes are required to report information, which is going to be captured to a sink, either in real-time or after its processing and storage, either way the operations are triggered by events.

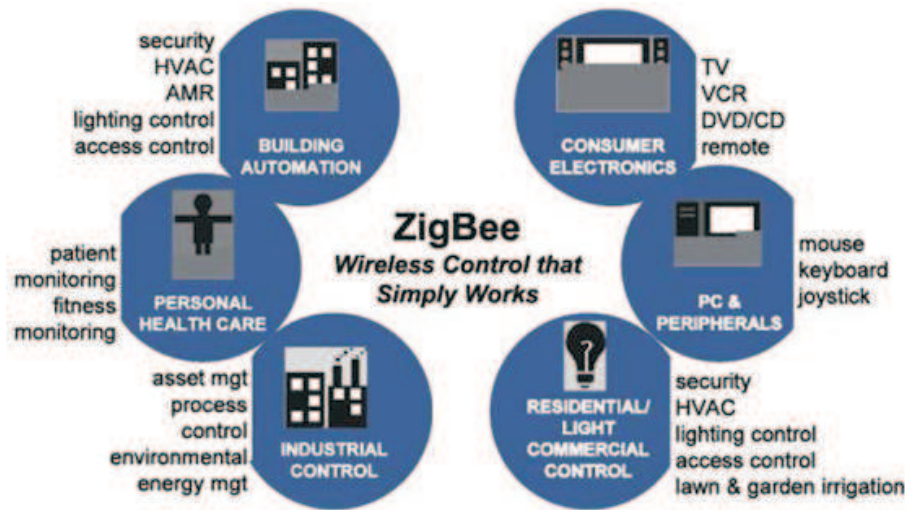


Figure 1.5 ZigBee Applications

- **Measurement and Actuation** In this scenario, both one-to-many and many-to-one models are involved. Nodes usually function on demand. For measurement applications, data processing is sometimes required.
- **Tracking** WSNs could work in MANET fashion, which means wireless sensor nodes are mobile in particular applications. On the other hand, the sources of information can also roam around within the area monitored by a group of static wireless sensor nodes.

However, we would like to classify WSN applications according to the specific purposes of them, for we think this taxonomy matches the design goals of WSN protocols.

- **Data Oriented Applications** In the large scale implementations of WSNs in monitoring and detection applications, information itself is usually much more important than the identity of the information sources, and the location or address of sensor nodes are unnecessary. Instead, the general information in the appointed area or the area around a target is needed. Furthermore, power consumption is the primary limitation of this type of application. Unnecessary communication is avoided due to data aggregation, and power efficiency is therefore improved. In this case, the collaboration among sensor nodes to process distributed raw data

can present an intuitional result and improve the accuracy of monitoring and measurement.

- **Data Source Oriented Applications** For WSN applications like AMR, office automation and stock management, each single node has its own task. Collected data need to be identified by the identity of sources. In addition, sensor nodes might act as actuators. End-to-end communication is the basis of the data traffic in this type of application.

1.3 GENERAL DESIGN PRINCIPLES AND CHALLENGES

It is evident from the last section that the design of WSNs is highly application-specific; however, the common traits of applications can lead to general challenges of WSNs design.

- **Topology Control** Most of current WSN applications show that sensor nodes are stationary after they are dropped or deployed in the target area. This is an attribute that distinguishes WSNs from other ad hoc networks. The pre-configured deployment requires topology management to improve the WSNs efficiency according to the characteristics of applications; on the other hand, auto-configuration is crucial for the wireless sensor nodes that are randomly dropped in the field. As wireless channel interference, power availability and nodes damage exist all the time, topology control appears to be indispensable for WSNs to operate in the highly dynamic environment.
- **Life Span** In many scenarios, the energy storage of a sensor node is very limited, which consequently limits the lifetime of the sensor node as well as the lifetime of the whole WSN. Ordinary AA alkaline cell and AA lithium batteries are still the power supply options for commercial applications [5]. Replacing flat batteries is usually unpractical or even impossible, especially when sensors are dropped in the field. All applications require WSNs to stay operational as long as possible

to accomplish expectant missions. Rechargeable batteries can extend the life-time of sensor nodes by obtaining energy supplement from the environment (sun, vibration and wind), but it requires a backup battery and an additional power converting circuit to provide consistent power supplies [2], which will apparently lead to the increase in both the size and the cost of sensor nodes. Hence, in this scenario, the improvement of energy efficiency and lifespan of WSNs becomes the primary design goal in this scenario; in addition, power-aware protocols and mechanisms are needed, and the trade-offs against quality of service (QoS) shall be considered [5].

- **Fault Tolerance** Sensor nodes failure might happen due to hardware damage or software errors. Running out of batteries and interruption of communication can also cut off sensor nodes from the whole network. Both increasing deployment density of sensor nodes and topology control can help a WSN as a whole to function correctly regardless of nodes failure.
- **Cost of Wireless Sensor devices** [5] To commercialize WSNs, the price of individual sensor nodes is the key concern. The target cost of a single sensor node is generally expected to be less than \$1, while the cost of a wireless sensor device based on Bluetooth technology is currently around \$10. Meanwhile, the size of sensor nodes tends to be smaller: four key components of sensor devices, a *power unit*, a *sensing unit*, a *processor* and a *transceiver* have to be fitted into a $2 \times 5 \times 1 \text{ cm}^3$ module, as shown in Figure 1.6 [9].
- **Standardization** As the development of WSNs, another issue, standardization, begins to attract people's attention due to its direct influence on the cost and the commercialization of WSNs. In fact, the cost-effective commercial applications are being encumbered with highly application-specific protocols. A straightforward design goal of WSNs is clearly given by Kazem, Daniel and Taieb (2007): "*The goal of WSN engineers is to develop a cost-effective standards-based wireless networking solution that supports low-to-medium data rates, has low power consumption, and guarantees security and reliability.*" (p19). Several standards are

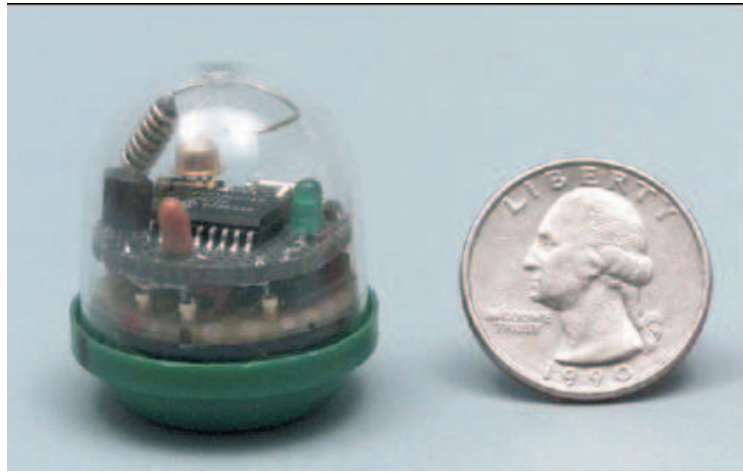


Figure 1.6 Sensor Web 1.0 pod, circa 1998 which includes antenna, battery, and sensors.

being implemented in the lower layers of WSNs: IEEE 802.15.4/ZigBee tend to be used in an indoor environment, while IEEE 802.11x and WiMax (IEEE 802.16) might be suitable for outdoor deployments. Since the networking layer is the characteristic that differs WSNs from traditional networks [5], WSN researchers are mainly focusing on the standardization of the routing protocols. Furthermore, standardization also covers various other issues that include the protocol stack design, cross-layer design, middlewares, on-board operating systems (OS) and operation softwares in remote sinks.

The large diversity of the applications always pushes the design of WSNs to a higher level. Quality of service (QoS), energy efficiency, scalability, robustness, self-configuration and data processing capacity are all expected to be optimized; however, none of current protocol cluster is capable enough to satisfy all of these requirements.

1.4 WIRELESS SENSOR NETWORK ROUTING PROTOCOLS

Routing is the act of directing and delivering data across a network from a source to a destination, whose kernel is path determination [8]. For wireless sensor networks, an efficient routing protocol is especially important. Firstly WSNs are ad hoc wire-

less networks, in which data needs to travel via a multi-hop path from a source to a destination; therefore, communication in WSNs involves more path discovery and determination. Secondly, the energy limitation of sensor nodes is the primary issue: sensing, actuator functions, data processing and communication are all accompanied by power consumption, among which the communication via wireless is the most power consumptive operation for a wireless sensor node [2, 10, 11]. Therefore a simple, robust, energy aware routing protocol should be the design goal.

1.4.1 Taxonomy of Ad Hoc Routing Protocols

Ad hoc routing protocols are mostly being developed in mobile manner, known as mobile ad hoc networks (MANETs). There are obvious differences between WSNs and MANETs. A WSN is generally composed of a group of sensor nodes, whose main task is to collect and exchange information. The communication between nodes works in three basic manners: one-to-many, many-to-one and any-to-any [8]. On the other side, the development of MANETs focuses on handling the mobility of handsets and the limited wireless transmission range. Therefore the routing protocol design should treat WSNs as a whole system to achieve certain tasks, while MANETs routing should deal with the problems of communication between any two or more mobile users [2].

As the basis of WSNs routing, ad hoc routing technologies have attracted researchers for years, and a large amount of protocols have been developed. One of taxonomy categorizes ad hoc routing protocols into *Table-driven* schemes and *Source-initiated* schemes [12].

- **Table-driven Protocols** Available paths across the networks are maintained at all times; however, such a benefit requires not only the constant broadcasting of routing information to update relative tables, but also much memory space to maintain the tables. This feature consequently leads to substantial data flooding and power consumption.
- **Source-initiated Protocols** Paths are intended to be discovered by a path

discovery process initiated by source nodes on demand. This kind of protocols are more energy efficient, because they released the resources used to broadcast and record various tables in table-driven protocols. However, a disadvantage is that a sender must wait until a path is discovered and built, which increases the latency. Furthermore, the route discovery involves messages flooding throughout the networks as well.

In addition, some *hybrid* routing strategies, combining table-driven protocols and on-demand protocols, are used in WSNs with cluster structures, where table-driven protocols are used for local communication within a cluster, and routes across clusters are discovered by on-demand strategies [5]. The range of ad hoc routing protocols is too wide to be fully covered here. A detailed overview of existing ad hoc routing protocols can be found in the survey papers [12, 13, 14].

1.4.2 Taxonomy of WSN Routing Protocols

The general ad hoc routing protocols with mass data flooding are obviously not ideal for WSNs. Under the push by WSN applications, the categories of WSNs are not ambiguous any longer, which requires the considerations on various system architecture issues, as listed below, when designing a WSN routing protocol.

- **Environment Dynamics** Most of WSN applications assume that sensor nodes are stationary after deployment, but the wireless environment itself is highly dynamic: links between nodes can be interfered easily, and sensor nodes could fail for various reasons and recover again after a while. The topologies in the *near-term* commercial applications, mentioned in the last section, are mostly predetermined; however, the structured topologies could still be fragmentized in dynamic wireless environment. Therefore, routing protocols should be robust enough to against the environment dynamics.
- **Energy Consumption** Energy is undoubtedly the principal consideration in

the WSN routing design. All routing strategies are trying to consume less energy on the basis of respective scenarios. Since transceiving is the most power consumptive operation in WSNs communication, improving data delivery efficiency and reducing retransmissions are the essential issues for WSN routing research.

- **Data Traffic and Delivery Models** The data traffic and delivery in WSNs highly depends on applications. The traffic is convergent in most scenarios, while some are any-to-any or many-to-one (e.g. in the *near-term* commercial applications). Data can be delivered to sinks in three ways: continuous, event-driven or query-driven [15]. Routing strategies can be optimized to suit the three data delivery models to achieve energy saving.

To address these design requirements, besides ad hoc routing, many routing schemes have been proposed specifically for WSNs, which are generally classified as data-centric, hierarchical, location-based and hybrid routing strategies offering multiple paths for flat network architecture [5, 16].

- **Data-centric Routing** When data is delivered through multiple paths, or multiple sensors generate same data from same monitored targets, redundancy thus to become significant; therefore, routing protocols are expected to implement data aggregation among a group of sensor nodes during data collection and delivery in a target area. The first routing protocol that reduces redundant data and power consumption by negotiation among nodes is SPIN, whose data advertisement mechanism prevents redundant data from being sent repeatedly, but does not guarantee the data delivery [17, 16]. Another milestone in the data-centric routing research is Directed Diffusion, which inspired many later routing protocols despite the limitation of the only query-driven data delivery model [18, 16].
- **Hierarchical Routing** In the networks with flat structures, some "gateway" nodes will be overloaded as the scale grows. The data congestion problem on these "gateway" nodes can be solved by either aggregating the data or distributing the load. Some routing protocols have been developed based on hierarchical

structures. They perform data aggregation and fusion by network clustering in order to decrease unnecessary transmission to the sink. LEACH [11] is one of the first and the most popular hierarchical routing protocol, whose idea of selecting and cycling the duty of cluster heads inspired many later hierarchical protocols. Its mechanism of delivering data along the cluster heads to the sink achieves a great reduction in energy consumption; however, the single-hop routing within the clusters limits the scalability of the protocol, and the duty cycling of cluster heads brings extra overhead [16].

- **Location-based Routing** Location-based routing protocols are primarily developed for MANETs to deal with end-to-end communication using geographical information. Since the proposal of the structured addressing scheme [19], addresses could be used to identify and locate sensor nodes in a network. However, location-based routing can still be applicable in some mobile WSN applications. For instance, the sink could be a fixed Base Station (BS) with a large transmission range, where sensor nodes roam in and out. The sensor nodes within the coverage of the BS can receive the location information of both themselves and their neighbors, and then they can use location-based protocols to communicate with the BS [20, 21].

1.5 ADDRESSING SCHEMES

When discussing routing technologies, the address structure and the naming system of networks can not be ignored, which are the fundamental components used to denote and find end users or messages in networking.

Addresses can be either bound with hardware during the manufacturing process or assigned on demand by addressing protocols. The on-demand addressing protocols can be further sorted into *centralized* and *distributed*. The first one employs a central addressing coordinator and the latter one does not respectively. There are currently two types of distributed addressing schemes, *flat* and *hierarchical* [22].

- Flat Addressing** The majority of MANET routing protocols employ a flat addressing mechanism. In a flat network, addresses are randomly assigned to nodes; therefore, the distribution of addresses is unrelated to the network topology. Although the addressing process is easy, a flat address structure always makes it hard to build a path between two arbitrary nodes.
- Hierarchical Addressing** A hierarchical addressing scheme can make the path discovery process easier due to its systematic structure. The distribution of hierarchical addresses also does not need a central coordinator; however, it causes the waste of address resources if the network grows irregularly. The ZigBee tree structure is such an example that is commonly used in WSNs. In addition, networks do not reserve addresses for the nodes roaming out of range. This feature makes hierarchical addressing more adaptable for static networks rather than for mobile networks.

Figure 1.7 illustrates the difference between hierarchical and flat address spaces [23].

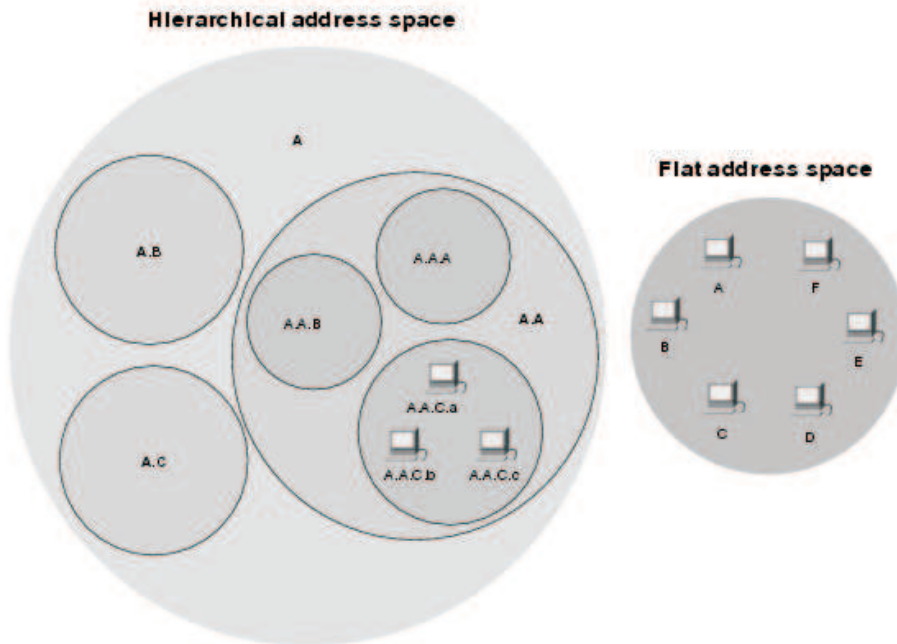


Figure 1.7 Hierarchical and flat address spaces differ in comparison operations

1.6 DISSERTATION OVERVIEW

As a member of ad hoc networks family, WSN technologies are largely derived from MANETs, especially routing protocols, many of them take into account specific properties of WSNs. The literature review on the WSN routing protocols in this work shows that addressing and naming are usually treated as unavailable resources in the WSN routing protocols, efficient data delivery thus to become the focus of routing research instead of robust path discovery and maintenance. Bhatti and Yue (2006) proposed an addressing scheme for multi-hop networks, which provides a systematic address structure for WSNs and can avoid the fatal node failure problem that could occur with the ZigBee tree structure [19]. This inspired our work in this thesis to utilize the systematic address structure to develop a new path discovery strategy for the WSN routing protocols.

The main contributions of this work are the following:

- We have studied various routing protocols proposed in past literatures for WSNs and the respective advantages and drawbacks. We defined the problem, and then addressed the research topic of this thesis.
- We studied and analyzed the structured addressing scheme proposed by Bhatti and Yue and applied the scheme to a network model of practical WSN applications.
- We proposed an adaptive flooding scheme that takes advantages of the systematic address structure, and applied our path discovery strategy in ZigBee routing protocol, thus, yielded a new routing protocol for static WSN applications.
- We evaluated the new proposed routing protocol in terms of message delivery ratio, redundancy and energy dissipation, and showed that our scheme under certain wireless channel circumstances is more energy aware and produces less redundancy than reactive ZigBee routing protocol. Compared with single-path

routing protocols, the newly proposed routing protocol improves message delivery ratio by adaptively increasing redundancy in the presence of link failures.

This dissertation begins with a general introduction of WSNs and ZigBee, followed by a summary of WSN applications and design challenges that are the guide-line of this work. The next section is a brief review of past literatures including routing protocols and addressing schemes. Chapter 2 gives a detailed discussion of ZigBee routing algorithm and Bhatti and Yue's addressing scheme, as this represents the foundation of the new routing protocol. In Chapter 3, we present our Address-Based Adaptive Flooding routing protocol (*ABAF*). The network model based on the systematic address structure and the adaptive flooding algorithm implemented in route discovery process are described in detail. The simulator, the simulation model and the results of the simulation are discussed in Chapter 4. The simulation compares *ABAF* with Ad-hoc On-demand Distance Vector (AODV) that is employed by ZigBee specification as one of the routing solutions in terms of message delivery ratio, redundancy and energy dissipation. The dissertation ends with a conclusion and possible future works in Chapter 5.

Chapter 2

PRIOR AND RELATED WORKS

As we discussed in the previous chapter, standardization is one of the necessary conditions for the commercialization of Wireless Sensor Networks. ZigBee is currently the most specific and well-maintained technical standard for low-cost low-power wireless networking. A simplified version of Ad-hoc On-demand Distance Vector (AODV) algorithm is employed by ZigBee Specification in the path discovery, maintenance and repair process [24]; therefore, it is also used as the benchmark in this work. Moreover, the new routing protocol is built on the structured addressing scheme proposed by Bhatti and Yue. Therefore, for this chapter, we are going to study and analyze the ZigBee routing [3] and the new addressing strategy [19] in detail.

2.1 ZIGBEE ROUTING DISCOVERY AND MAINTENANCE ALGORITHMS

2.1.1 Route discovery

In ZigBee specification 2007, both multicast and broadcast are included in the routing strategies; therefore, the route discovery is performed for three situations in a ZigBee network.

- **Unicast Route Discovery** To find and establish a route between a particular source node and a particular destination node.
- **Multicast Route Discovery** To find and establish routes between a partic-

ular source node and a multicast group with a common multicast group ID, for example, a cluster.

- **Many-to-one Route Discovery** To find and establish routes between a sink device to all ZigBee routers and ZigBee coordinators within a given radius.

Here in the thesis, only the content of basic unicast routing in the original ZigBee specification is considered; furthermore, we also neglect the routing along the ZigBee tree structure applied while possible in ZigBee specification, because it does not adapt general network topologies.

Initiation of Route Discovery

The *Path Discovery* process (see Figure 2.1) shall be initiated by the NWK layer of a ZigBee router or a ZigBee coordinator, either on receipt of a ROUTE_DISCOVERY request generated by the higher layer, for which there is no routing table entry corresponding to the destination address, or when a frame is passed in by the MAC sub-layer, for which the destination address in the routing header is not the address of the current node, the address of a valid one-hop neighbor or a broadcast address, and there is also no routing table entry corresponding to the destination of the frame. The path discovery is initiated by broadcasting a *route request* (RREQ) message to neighbor nodes. The following important fields are contained in the route request overhead:

- *Source Device Address*
- *Destination Device Address*
- *Route Request ID*
- *Source Sequence Number*
- *Destination Sequence Number*
- *Hop Counter*

In either case above, a corresponding routing table entry shall be established and set to DISCOVERY_UNDERWAY. If the corresponding routing table entry exists, but the status is not ACTIVE, then the entry shall be set to DISCOVERY_UNDERWAY as well.

Meanwhile, the nodes that generate route requests shall maintain a request counter as route request identifiers. The counter is incremented as a new route request is created. A *route request identifier* and a *source device address* are used as an unique identification (ID) of each route request frame. An intermediate node shall drop redundant request frame with the same ID. Each route request item in routing table is set with an expire timer. The expired request entries with the status of DISCOVERY_UNDERWAY shall be deleted. Once a route discovery request entry is created, the route request command frame is passed to the MAC sub-layer for broadcast. A route discovery request is allowed to be retried by the NWK layer for *nwkInitialRREQRetries* times after the initial broadcast of the route request frame.

On Receipt of a Route Request

A node shall drop the route request frame if it is the destination; otherwise, it is an intermediate node and shall determine if it has routing capacity.

- **Definition 2.1:** *Routing Capacity* - A ZigBee router or ZigBee coordinator maintains a routing table as well as a route discovery table. Routing table entries are long-lived, while route discovery table entries last only as long as the duration of a single route discovery operation and may be reused. Routing capacity means both routing table capacity and route discovery table capacity.

In ZigBee specification 2007, the intermediate node without routing capacity proceeds the route request according to ZigBee tree routing.

If the intermediate node has routing capacity, it shall check if a route discovery table entry with the same ID and source address exists. if no such entry exists, then one

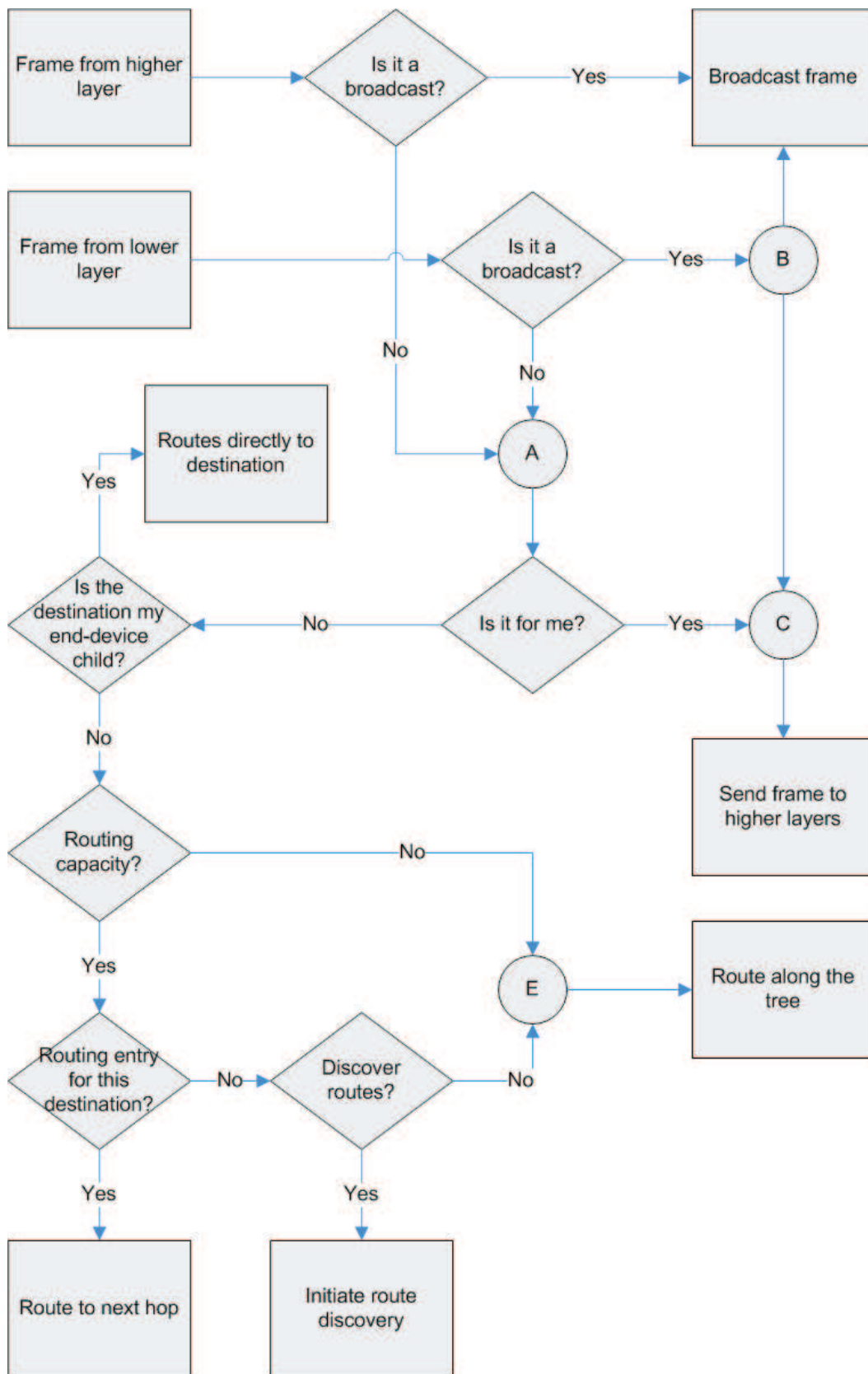


Figure 2.1 Basic ZigBee Route Discovery Algorithm

shall be created and set to expire in *nwkRouteDiscoveryTime* milliseconds. Then, the node shall check the corresponding routing table entry with same destination address. If the routing table entry is not ACTIVE or does not exist, one shall be created and set to DISCOVERY_UNDERWAY. The route request shall then be rebroadcasted. Otherwise, the destination sequence number in the routing table entry and the route request respectively shall be compared. If the route request has a greater destination sequence number, it shall be forwarded. If either the node is the intended destination or the destination sequence number in the current routing table entry is greater, a *route reply* (RREP) command frame shall be generated and replied along the *reverse path*. The route reply frame shall contain the following information:

- *Source Device Address*
- *Destination Device Address*
- *Expire Timer*
- *Hop Counter*
- *Destination Sequence Number*

The reverse path is automatically set up by the route request. As the route request message travels from a source to a destination, each node on the path records the address of its upstream neighbor node, from which the first copy of the route request was received. Each reverse path entry maintains an expire timer, which shall be set at least long enough for a corresponding reply message is generated and reaches the source device.

If a route discovery table entry with the same ID and the source address exists, the respective link cost shall be compared. The one with lower link cost shall be put in the route discovery table. Figure 2.2 illustrates how to process a route request.

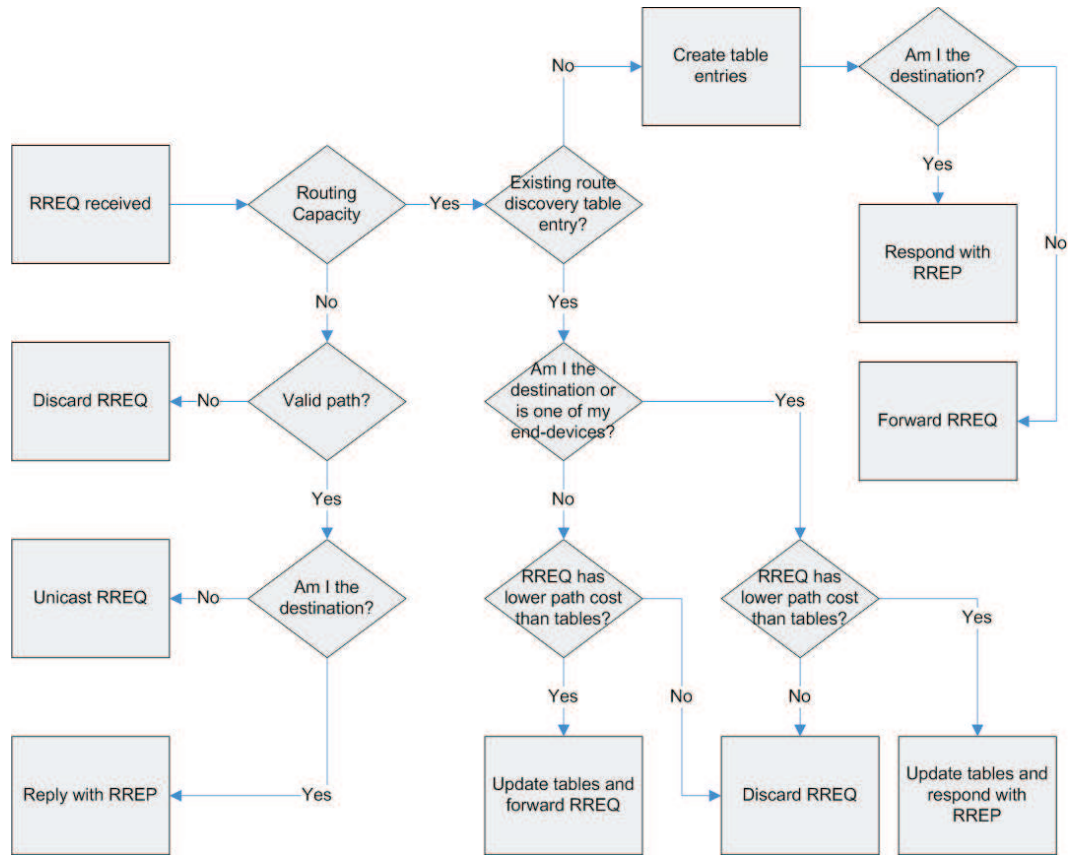


Figure 2.2 Receipt of Route Request

On Receipt of a Route Reply

In ZigBee specification 2007, the intermediate node without routing capacity proceeds the route reply according to ZigBee tree routing.

If the intermediate node has routing capacity, it shall check whether it is the destination of the route reply frame. If positive, it shall search its route discovery table for the entry corresponding to the route request ID in the route reply frame header. The route reply frame shall be dropped, if a route discovery table entry does not exist. Otherwise, the node shall search its routing table for an entry with a destination address same as the responder address in the route reply command frame. If no such routing table entry exists, the route reply shall be discarded, as well as the corresponding route discovery table entry. If the routing table entry exists and has the status of DISCOVERY_UNDERWAY, it shall be set to VALIDATION_UNDERWAY. The last hop of the route reply frame shall be set in the next hop field in the routing table entry. The path cost recorded in the route reply frame shall be inserted in the routing table entry. If the status of the routing table entry has been ACTIVE or VALIDATION_UNDERWAY, the node shall compare the path cost in the route reply command frame and the routing table entry, then, either update the routing table entry by the smaller one in the route reply frame or discard the route reply with a larger path cost.

If the device is not the destination of the route reply, it shall repeat comparison process above and update corresponding information field. If the route entry is updated, the node shall search the route discovery table for the next hop address corresponding to the route reply, compute the link cost, update the route reply frame and forward the route reply to the destination. Figure 2.3 illustrates how to process a route reply

2.1.2 Route Maintenance

The NWK layer maintains failure counters for outgoing links to the neighbors, to which it sends data. If an outgoing link fails, a failure counter value shall be generated, or

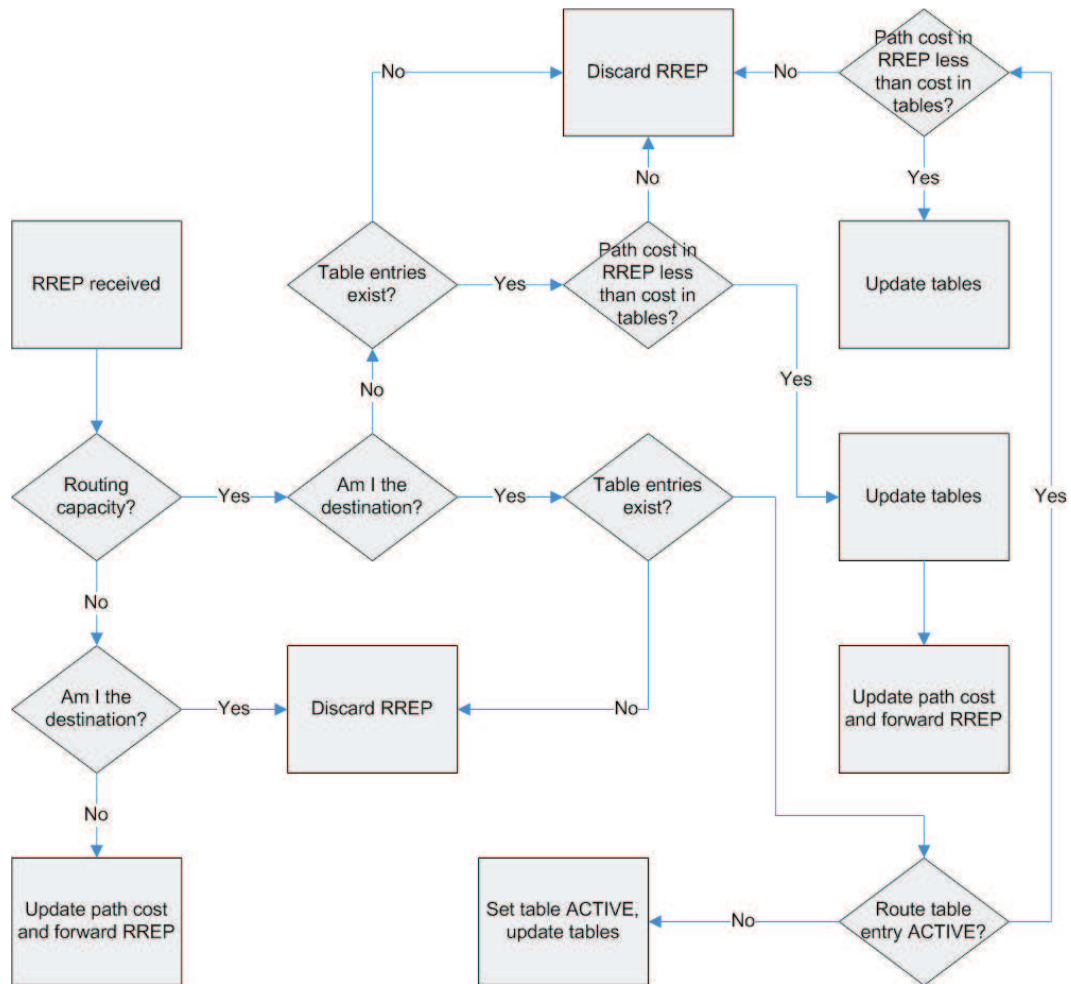


Figure 2.3 Receipt of Route Reply

a time-windowed scheme shall be employed. Repair operations are highly unrecommended, because they may flood the network and cause traffic disruptions.

If a link failure happens while the node is transmitting a message via an unicast route, the node upstream of the broken link shall send a message with a network status frame back to the source to indicate the reason for the failure, and a NLME_NWK_STATUS indication shall be passed to the next higher layer to indicate the link failure at the same time. On receipt of a network status command frame indicating a link failure, the node shall remove the routing table entry corresponding to the original destination of the forwarded messages, and also inform the next higher layer of the failure using the same NLME_NWK_STATUS indication. Upon receipt of a link failure notification, the original node shall restart a path discovery process if the path is still required.

2.2 A STRUCTURED ADDRESSING SCHEME

As we have introduced in the first chapter, ad hoc networks require a set of protocols to assign addresses and names to nodes dynamically. A flat addressing scheme has an easy address assigning process. But much Flooding of address notifications is generated due to the random address selection and synchronization of address tables throughout the network. Moreover, the irregular and unsystematic address structure makes it hard to orient messages between any two arbitrary nodes [25, 26]. Although hierarchical addressing algorithms suit static WSN applications better, they are still suffering from respective defects. The concept of *Binary Split* in [27] deals with the problem of address resource waste, caused by irregular growing of networks, by borrowing addresses from other nodes when a node used up its own. However, this behavior makes the blocks of addresses noncontiguous, which means the address pattern is not systematic any longer. ZigBee tree structure employs a parameter d , called depth in the network, to limit the number of child nodes for each node. This parameter is pre-configured and fixed during the network growing, which ensures the even allocation of address pools and the systematization of address patterns. ZigBee tree does not share addresses

between devices, so it happens that one parent node exhausts its address pool while other parents have addresses unused. A new device will not be able to join the network if no parent node with spare addresses is within its transmission range [3]; in addition, the waste of address space still exists in the irregularly growing networks [19].

2.2.1 Scheme Description

Bhatti and Yue have recently proposed a new addressing scheme [19]. This scheme treats the entire address space as a n -dimensional hyper-cube. Figure 2.4 shows a 3-dimensional address cube.

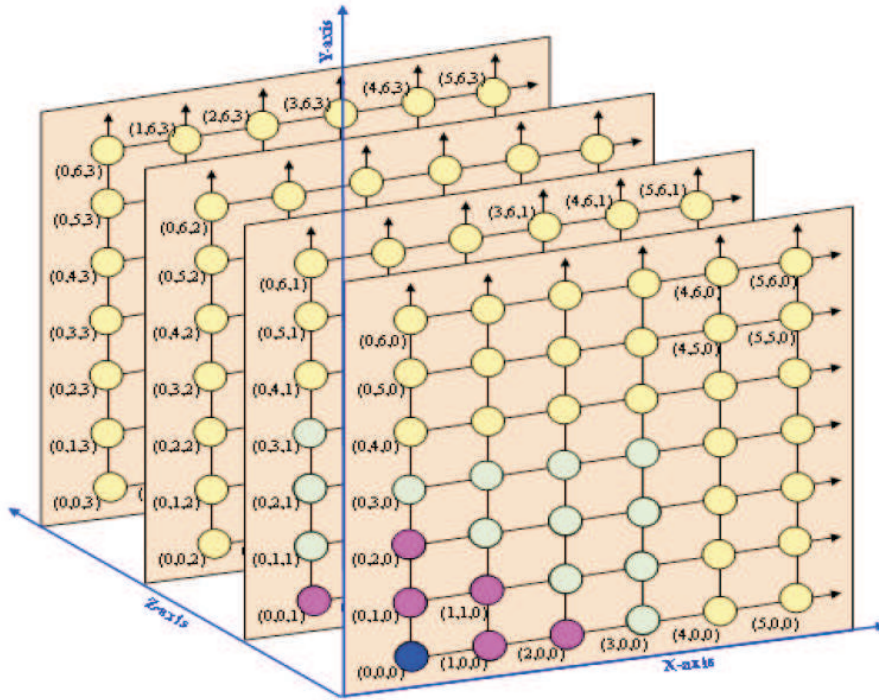


Figure 2.4 3-dimensional Hypercube Structure

The following annotations should be aware:

- **Hyper-Cube** The regular shaped cube, for example the 3-dimensional cube in Figure 2.4, is only a logical structure that does not reflect the exact physical position of nodes, also see Figure 2.5 and 2.7.

- **n -dimensional** In the n -dimensional structure, each intersection represents a node that has a Cartesian coordinate as its address. n is also the number of address components that is a pre-configured parameter. For example, if n is 3, the address will be (x, y, z) ; hence, a node in the n -dimensional cube can have a max of n child nodes, which are $(x+1, y, z)$, $(x, y+1, z)$ and $(x, y, z+1)$ for the given example, also see Figure 2.4.
- **Address Structure** The new address assigning scheme provides hierarchical systematic address patterns.

The newly proposed addressing scheme states such advantages and advancements:

- 1) Addresses are allocated in a systematic manner. The address space is regular and hierarchical.
- 2) The waste of address resource is reduced. Even if the physical location of nodes was not uniformly distributed, the mechanism would still try to assign addresses in a regular n -dimensional pattern.
- 3) No flooding of notification of assigned addresses throughout the networks, and no flooding of address tables.
- 4) The robustness against network partitioning due to parent nodes failure is enhanced.
- 5) The restriction on the depth of address hierarchy in ZigBee Tree structure is removed.

For the simplicity of illustration, we consider $n=2$ here; thus the hyper-cube takes the form of a plane grid pattern shown in Figure 2.5. Each node has an address with two components as (x, y) . If the address space in the given 2-dimensional network, for instance, is 16 bits long, the two components can be assigned 8 bits each or 10 bits and 6 bits respectively or any other pairs. The maximum value of each address component

in the first case is 2^8 ; hence, the address range is from $(00, 00)$ to (FF, FF) . A network starts at the origin point of the Cartesian coordinates, $(0, 0)$, and grows along any axis corresponding to the physical distribution of the nodes.

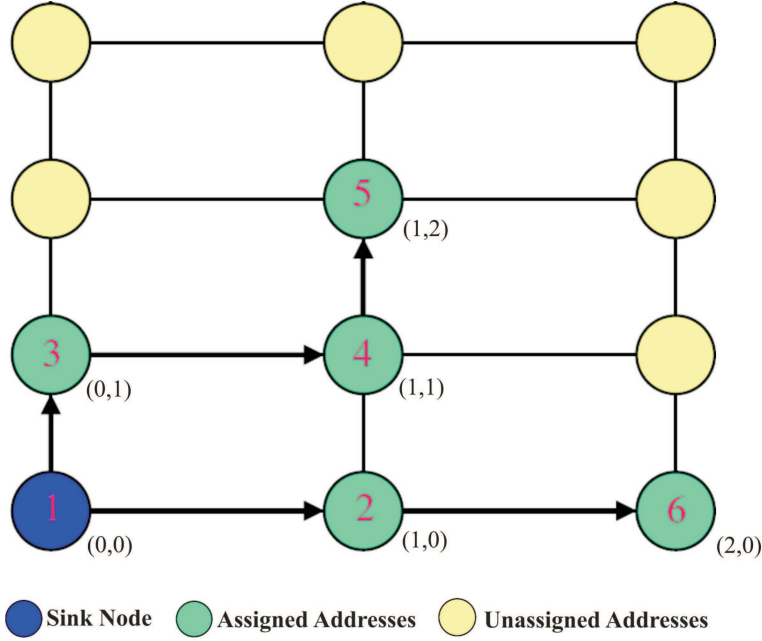


Figure 2.5 Addressing Process in A 2-dimensional Address Grid. The numbers in nodes denote the order that nodes join in the network, while the arrows represent the successive relationship of addresses.

The addressing process is illustrated in Figure 2.5 and works by following the steps below:

- Step 1)** We assume both of the address components are 8 bits long. The origin node, say N_0 , could be not only $(00, 00)$ but also any of the other three corners of the grid, $(00, FF)$, $(FF, 00)$ and (FF, FF) . $(00, 00)$ is used here for convenience.
- Step 2)** A parent node only has the addresses, which are adjacent to its own address, available. The corresponding address components of the new address are either greater or equal to the parent address. For example, the first node N_0 has a new child node N_1 joined the network. The addresses available to assign to N_1 are $(00, 01)$ and $(01, 00)$.
- Step 3)** Once a node is assigned an address by one of its potential parent nodes, all the other potential parent nodes shall be informed so that address conflicts can be avoided. In a 2-dimensional network, a node is assigned an address $(x,$

y) by node A $(x-1, y)$, the new node is now responsible for sending a message to the other potential parent node B $(x, y-1)$ to claim that the address (x, y) has been occupied. In general, a new node in a n -dimensional address space theoretically has n potential parent nodes, but only one is going to build the actual relationship with it, and then the other $n-1$ potential parent nodes need to be notified. In Figure 2.5, when node 4 joins in the network, both node 2 and node 3 could be its parent node. If node 4 is assigned an address $(1, 1)$ by node 3, it must send a message to node 2 to claim the occupation of the address $(1, 1)$.

The address pattern would grow irregularly into a chain or a random shape, if addresses were assigned along a single axis. Figure 2.6 gives an illustration.

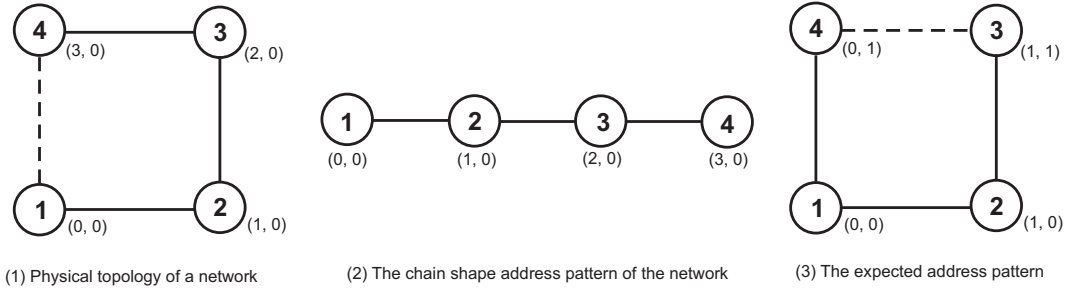


Figure 2.6 An Example of A Chain Address Structure. The number denotes the order that nodes join in the network. The dash lines indicate possible links between nodes. As we can see, although the physical locations of nodes are in a grid pattern, the address structure grows in a chain by assigning addresses improperly.

A simple strategy is suggested by the authors to ensure a more uniform address distribution. If both address components are either even or odd, the next address to be assigned should be along one available axis; otherwise, the address should be assigned along the other axis. For example, node $(2, 2)$ and node $(3, 5)$ will try to assign $(3, 2)$ and $(4, 5)$ to their child nodes respectively, and node $(1, 2)$ and node $(2, 3)$ will try to use $(1, 3)$ and $(2, 4)$.

2.2.2 Discussion

Despite all the advantages, problems and defects still exist in the new addressing scheme. Some issues arose while we applied it. Because these issues do not cause much impact on this work, we are not going to discuss them in detail in this thesis. They are listed below.

- The scheme claims to be able to adapt any forms of ad hoc networks; however, the practical implementation shows that it is not flexible and even not feasible in highly dynamic environment like MANET, where nodes join in and roam out randomly, see Figure 2.7. It would work well if addresses were allocated after the deployment of nodes.
- The scheme does not advice whether addresses will be reserved for nodes partitioned from parents, and how to reserve the addresses if the answer is positive.
- The method to prevent networks from growing irregularly was proposed by the authors; however, applying it in a n -dimensional structure could be a problem when the number of address components, n , is bigger than 2.
- Both the pre-configured n and the start points on the corners of an address structure have a negative impact on the flexibility of the addressing scheme.

In an overview of the new addressing scheme, it does provide a more coherent address structure as the basis for routing protocols in multi-hop networks, especially in static environment; furthermore, it is more robust to network fragmentation, compared with ZigBee tree structure. The merits of the new addressing scheme is summarized in Table 2.1. Based on the analysis above, the proposed routing protocol in this thesis is assumed to work in such a 2-dimensional hyper-cube scenario, where the addresses are pre-configured according to the new addressing scheme, and stay constant once assigned. It will be proven in the following chapter that even a randomly deployed network topology is able to be fitted in a regular address structure, as long as the nodes

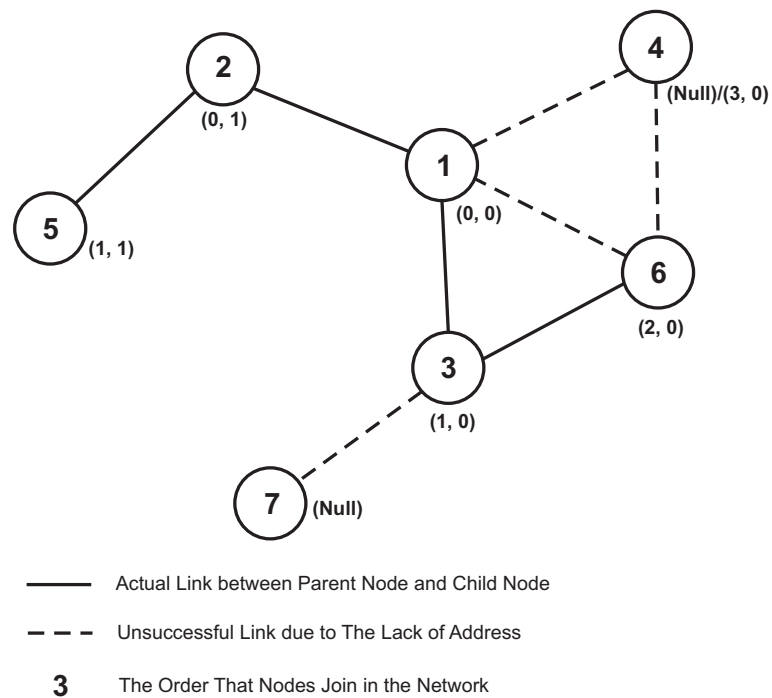


Figure 2.7 Disordered Address Assignment. Node 1 used up its addresses and have no address available for node 4. (1, 1) was assigned to node 5 according to the uniform address distribution strategy, which leads to the failure of the address assignment to node 7 by node 3. Node 3 and 5 have no physical connection, although the addresses are coherent; however, the opposite situation happens to node 1 and 6. Node 4 might obtain an address from node 6 later on.

deployment is done before the addressing; hence, the addressing scheme primarily suits our scenario for static WSNs applications.

Table 2.1 Comparison of Addressing Schemes

	Flat Addressing Schemes	Hierarchical (Binary Split)	Hierarchical (ZigBee Tree)	The New Scheme
Addressing Process	Easy	Medium	Medium	Medium
Flooding Overhead	High	High	High	Low
Address Structure	Irregular	Irregular if addresses borrow among nodes	Systematic	Systematic
Waste of Addresses	Low	Reduced by addresses borrow	High in irregularly growing networks	Low
Robustness against Network Fragmen- tation	Low	Low	Low	High

Chapter 3

ADDRESS-BASED ADAPTIVE FLOODING (ABAF) ROUTING PROTOCOL

3.1 MOTIVATION AND PROBLEM DEFINITION

The state of art in WSN routing protocol research was briefly introduced in the first chapter. The traditional routing algorithms for ad hoc networks are usually categorized into PROACTIVE and REACTIVE; meanwhile, protocols specialized for WSNs are proposed in terms of data centric, direct diffusion, hierarchical, geographical routing. Although some of them are Internet Engineering Task Force (IETF) protocols, to the best of our knowledge, none of them are employed by the *near-term* commercial WSN applications as an industrial standard. The routing algorithm specified in the ZigBee specification is a simplified AODV combined with a table-driven strategy and tree-structural routing [3].

Traditional routing protocols considered network dynamics more. To ensure networks are consistently operative, overhead increases with the increment of network size and network dynamics, which can easily exhaust network resources. The WSN protocols like data-centric routing and hierarchical routing are designed to achieve optimal data traffic with lower energy cost; however, the routing strategy in terms of path discovery seems to be overlooked in these routing protocols. The working environment is tough for WSNs: a single node may know nothing about the network at the beginning of its joining in (see Figure 3.1); whereafter, the limited energy is needed to keep sensing and updating the network topology. Under such circumstances with link failures, node

failures and channel interferences, routing is no mere a "checker game" with a clearly visible "chessboard", but a more complicated "labyrinth game". To survive in this tough environment, the scenarios of most WSN routing schemes require much extra information that is exchanged by flooding throughout the network. The information like node locations, device status (e.g. power level) are all acquired at the cost of overhead or other energy consumptive options (e.g. GPS in geographical routing). Some massive flooding could cause traffic congestions, for example, the synchronization of routing tables and address tables.

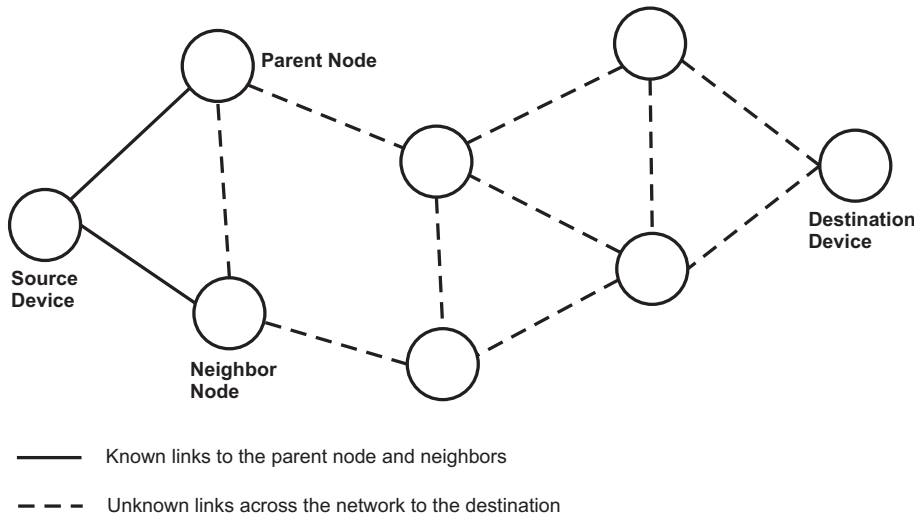


Figure 3.1 A new joining node knows nothing about the network except its parent node and neighbors. The routing protocol design should be from the point of view of a node not an observer.

Two simple routing rules are flooding and single-path routing. Flooding forwards a message to all neighbors until it reaches the destination. As long as the source and the destination node are accessible to each other, the message delivery is guaranteed. Looping is avoided by dropping redundant messages received by intermediate nodes. A time or hop threshold is set as an overhead to avoid endless propagation of messages. An alternative way is to forward packets to an arbitrary neighbor or a known neighbor closer to the destination. This results in the packet traveling across the network to the destination along a random path or a shortest path, and the packet delay could be consequently larger. As we saw in the last chapter, the ZigBee routing protocol floods networks by route request command frames in the route discovery phase. The

cost of maintenance can be low in a comparatively constant wireless channel, but the robustness decreases in an dynamic environment where link failure and node failure exist. Flooding and single-path routing are two extreme cases of routing methods. Although they are simple, their performances in terms of energy efficiency, delay and success ratio are likely to be poor.

On all accounts, nodes have to consider how to find a balance between performance and energy-efficiency [2]; therefore, new routing strategies are required to be capable of effectively managing the trade-off between optimality and efficiency [5]. Based on this standpoint, a preparatory strategy is proposed, which generates a partial flooding targeting the direction of the destination, and then takes advantage of shortest-path algorithms to reduce the redundancy in path discovery phase. The partial flooding and the shortest-path algorithms are developed based on the systematic address pattern. A parameter is set to control the partial flooding by changing the number of hops of flooded packets so that the whole route discovery process can be adjustable and flexible to adapt various wireless environments and achieve a satisfied balance between performance and energy efficiency.

3.2 SCHEME SCENARIO

3.2.1 Network Model

For *near-term* WSN applications, the sensor nodes are stationary after deployment, for example, a vineyard irrigation and monitoring system, a stock or dock management system. when containers can be moved and the irrigation devices in the field might be damaged, the network environment can be dynamic. The network model of our Address-Based Adaptive Flooding *ABAF* routing protocol is such a static network operating in a dynamic wireless environment.

- **Definition 3.1:** *Static Network* - A static network is a network that all its communication units are fixed after deployment, and immobile while communicating

with each other.

- **Definition 3.2:** *Dynamic Network* - A Dynamic network is a network with link failure and node failure that can cause local topological changes.

A single node failure makes all its incoming and outgoing links disconnected; therefore, for the convenience of analysis, only link failure is considered in the scenario.

In the *near-term* WSN applications, data delivery modes are usually event-driven or query-driven, and data traffic is mainly one-to-one and many-to-one (one-to-many). To improve the end-to-end route discovery efficiency, data traffic runs in the one-to-one mode in the scenario.

- **Definition 3.3:** *Event-driven* - Data transmission is triggered by local events, for example, the temperature or humidity level in soil.
- **Definition 3.4:** *Query-driven* - Data transmission is triggered by a query from sink(s). For instance, A sink can send a command to execute irrigation or a request for a report of freight information of the corresponding containers.
- **Definition 3.5:** *One-to-one* - Communications between any two arbitrary nodes. The process of message delivery could involve the aid of intermediate nodes.
- **Definition 3.6:** *Many-to-one* - The many-to-one mode is assumed to be composed of many one-to-one communications. Data aggregation is not considered in many-to-one mode in the route discovery phase.

Bhatti and Yue's structural addressing scheme is employed in the *ABAF* routing protocol as the basis of the network model. The address model follows the assumption below:

- **Assumption 1:** The address pattern of the network model is a 2-dimensional Bhatti and Yue's hyper-cube, a planar orthogonal grid. Each intersection in the

grid has an unique Cartesian coordinates. The address space starts from the origin $(0, 0)$ that can be any of four corners of the Cartesian coordinates grid. The size of the address space is $(N+1) \times (N+1)$.

In WSNs, sensor nodes are usually deployed in a certain density to ensure system reliability. The redundant deployment also makes it possible that nodes can always be divided into a group of four neighbor nodes to form a quadrangular grid in any cases, see an example shown in Figure 3.2. It can be deduced that Bhatti and Yue's structural addressing scheme can fit in any random 2-dimensional topology. What needs to be emphasized here is that the address pattern is a logical image, while the topology of the network model is a physical structure. Based on the conclusion above, to generalize the network graphs of various applications, the following practical assumption on the physical topology of the network model is made.

- **Assumption 2:** The physical topology of the network model follows exactly the same shape as the logical pattern of the address space.

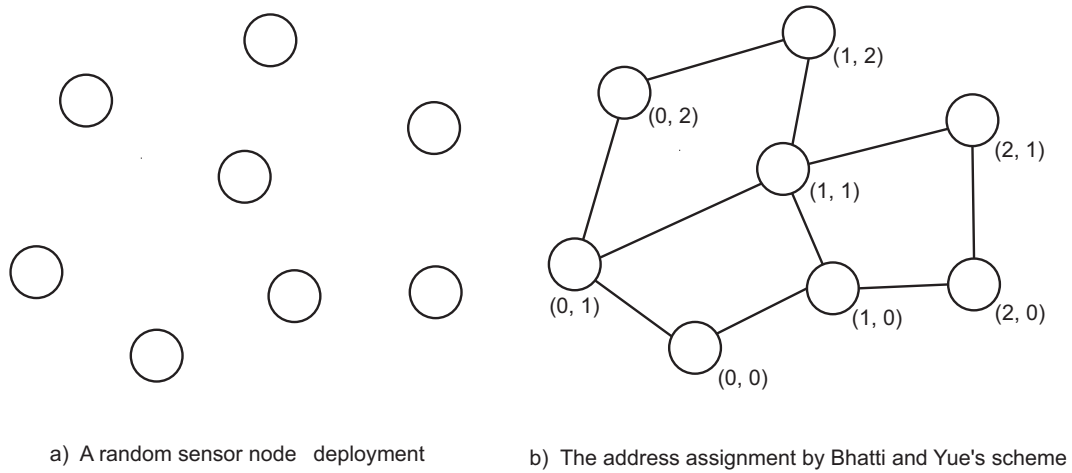


Figure 3.2 Implementing Bhatti and Yue's Addressing Scheme in an arbitrary topology

Therefore, both the address pattern and the physical network topology are assumed to be the graph shown in Figure 3.3.

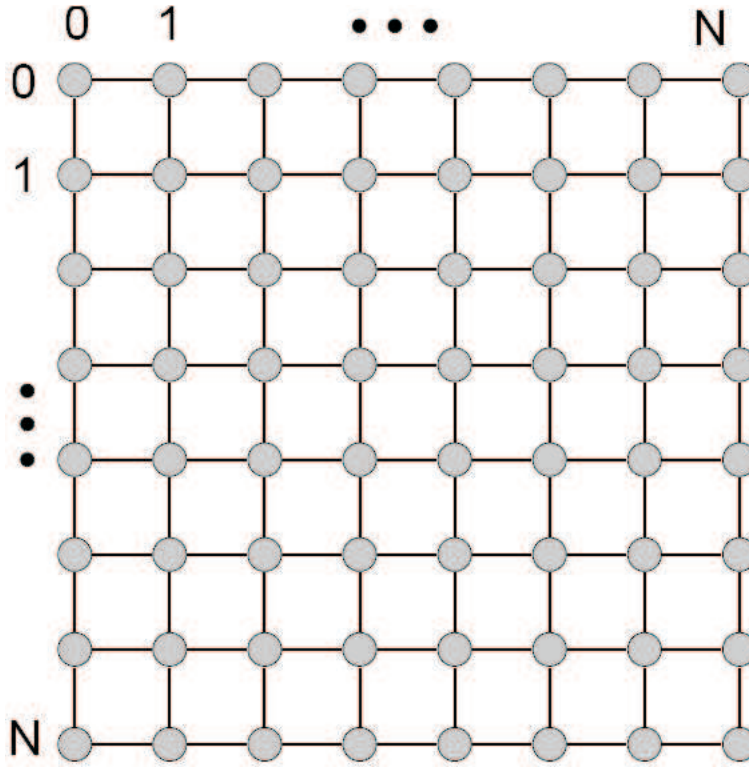


Figure 3.3 The *ABAF* Network Model. The topology of *ABAF* network model, as well as the address structure.

As noted above, the work focuses on investigating the communication between any two arbitrary nodes, assumed to be the origin $(0, 0)$ and the end point (N, N) . The network model can actually be an arbitrary section extracted from a larger network, as shown in Figure 3.4.

3.2.2 Network Configuration

The deployment and addresses assignment of sensor nodes are pre-configured following the network model shown in Figure 3.3. Miscellaneous **Network Configuration Assumptions** are summarized as below:

Assumption 3:

- 1) For the purpose of simulating network dynamics, a node failure equals to multiple link failures; therefore, only the link failure between two neighbors with a fixed

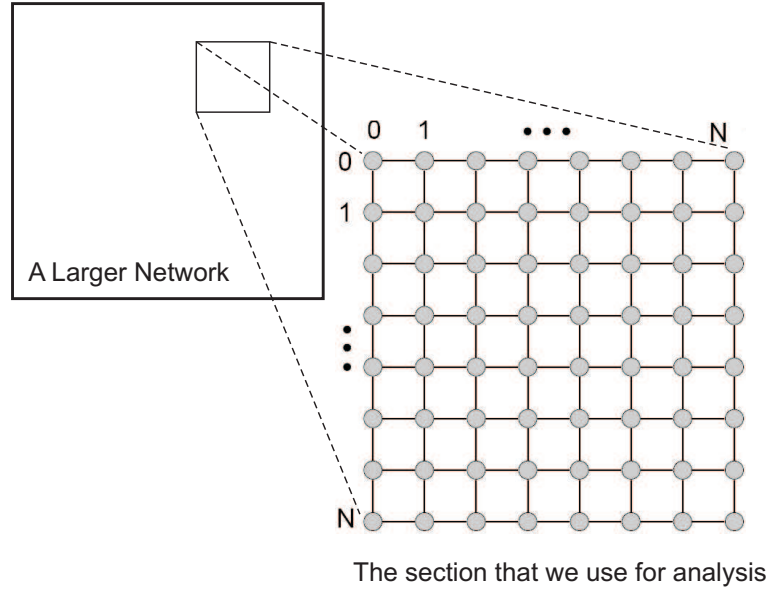


Figure 3.4 We can utilize an arbitrary section in a network for our analysis.

probability is considered, which is given by p .

- 2) The network model is composed of $(N+1) \times (N+1)$ sensor nodes as shown in Figure 3.3. No more nodes roam in or out during the simulation.
- 3) The density of sensor nodes deployment is high enough so that the transmission range of each node can cover all of its one-hop neighbors.
- 4) The distance between neighbor nodes are same due to the pre-configured topology; thus, every hop takes same time.
- 5) Direct communications can only occur between nodes that are directly linked, which is for the convenience of simulation.
- 6) All events happen in a time order in the simulation; Therefore, the wireless environment shall be kept still once a message exchange begins, which means the transmission power shall not be changed and no more nodes or links fail during a message travels to the next hop.
- 7) An end-to-end routing discovery process from $(0, 0)$ to (N, N) is investigated in this work.
- 8) All sensor nodes are capable to finish any assigned tasks, which means no node failure due to power exhaustion. The memory of each sensor node is redundant enough to record any tables and messages.

- 9) The energy dissipation of transmitting and receiving are approximately assumed to be equal, although the latter is higher in many practical WSNs [2].

3.3 SCHEME DESCRIPTION

AODV has a redundant route discovery process of flooding RREQ, which "overkills" static WSNs, while shortest-path routing is obviously less robust in a dynamic environment. The description of the *ABAF* routing protocol starts from implementing shortest-path routing and flooding in the defined network model. Thereafter, the impacts on shortest-path routing by flooding in the presence of link failure is investigated. A hybrid adaptive approach for static WSN applications is eventually proposed.

We state the following four main merits of the new proposed the *ABAF* routing scheme.

- 1) Fit WSNs with a systematic address space. Addresses become practically available resources for the WSN routing.
- 2) The path discovery process is adaptive. A controllable parameter is employed to keep the balance between energy efficiency and QoS. Less redundancy is generated while achieving the same latency as AODV.
- 3) Lower redundancy. For no extra information like nodes locations or the duty of cluster heads is needed, much less broadcast of exchanged information is involved; in addition, the size of overheads is also kept small.
- 4) Lower cost of route maintenance. Due to the merit of the systematic address structure, the size of routing tables and address tables can be reduced dramatically.

3.3.1 Shortest Path Phase

The scenario and the network model have been successfully defined in the previous section. In such a topology, if link failures are ignored, routing is no more than forwarding messages toward the direction of the destination orderly following the addresses. The process can be described as the steps below.

Step SP1) A message from a source node $(0, 0)$ arrives at an intermediate node (x, y) , for example. The destination of the message is $(x+n, y+n)$.

Step SP2) The intermediate node shall compare each corresponding address component of the destination with its own address. Here, the compared pairs shall be respectively x and $x+n$, y and $y+n$.

Step SP3) The result of the comparison shall lead to a decision on the next hop to the destination $(x+n, y+n)$. In this case, three choices are available, $(x+1, y)$, $(x, y+1)$ and a random choice among these two.

The repeat of the path decision process will finally lead to three simple routing methods, all of which produce a shortest path, see Figure 3.5.

- **X-Y Routing** A message from the source node $(0, 0)$ is forwarded along X -axis or Y -axis until it arrives at the edge $(x+n, y)$ or $(x, y+n)$; thereafter, it will reach the destination node $(x+n, y+n)$ along the other axis.
- **Zigzag Routing** Messages travel in a zigzag pattern. When they can not go in zigzag fashion any longer, they go along the appropriate axis to the destination.
- **Random Walk** Messages choose their next hop along X or Y direction randomly. If they reach an edge ahead of time, they are routed to the destination along the appropriate axis as above.

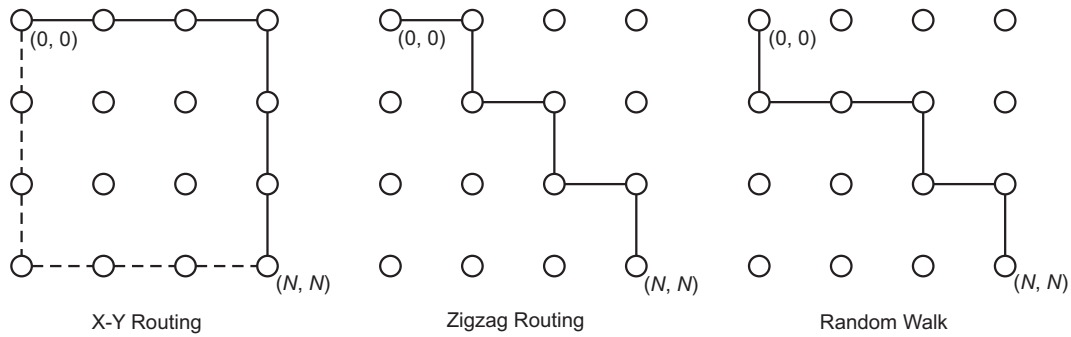


Figure 3.5 Shortest-path Routing. Three shortest-path routing methods based on the systematic address structure

3.3.2 Partial Flooding

As we stated, the shortest-path routing algorithms are based on the scenario in the absence of link failure. It obviously consumes the least resource to deliver data between two nodes; on the other side, it has the lowest robustness against network dynamics. In the systematic address structure, the three shortest-path routing methods orient messages according to the addresses, so does flooding.

Step PF1) A message to the destination (N, N) is generated by the APP layer of the source node $(0, 0)$ and passed to the NWK layer for broadcasting.

Step PF2) Upon receipt of the message, the NWK layer shall compare each corresponding address component of the destination with the source address $(0, 0)$. Here, both of the compared pairs shall be 0 and N .

Step PF3) The result of the comparison shall lead to a decision on the direction of flooding. As each node has maximum four links along x and y axis. Broadcasting from a node (x, y) thus to be divided into four multicast sections, $(x+, y+)$, $(x+, y-)$, $(x-, y+)$ and $(x-, y-)$. In this case, the message shall be set as a multicast packet with the multicast address $(x+, y+)$, and flooded to the next hop neighbors, both $(0, 1)$ and $(1, 0)$.

The multicast messages shall be flooded within the configured section and finally reach the array where the destination node (N, N) is. The partial flooding is illustrated in Figure 3.6.

The partial flooding fills the triangle area in between $(0, 0)$, $(0, 2N)$ and $(2N, 0)$. All nodes on the border of $(0, 2N)$ to $(2N, 0)$, where the destination node (N, N) is, satisfy the equation below.

$$(X_N - 0) + (Y_N - 0) = (N - 0) + (N - 0) = 2N \quad (3.1)$$

incoming links are broken at worst. Then $(x_d - x_s) + (y_d - y_s) + 1$ hops can reach all four neighbors of the destination node, and one more hop can deliver messages to the destination via any of the other two links if the node is still accessible. Figure 3.7 gives an illustration.

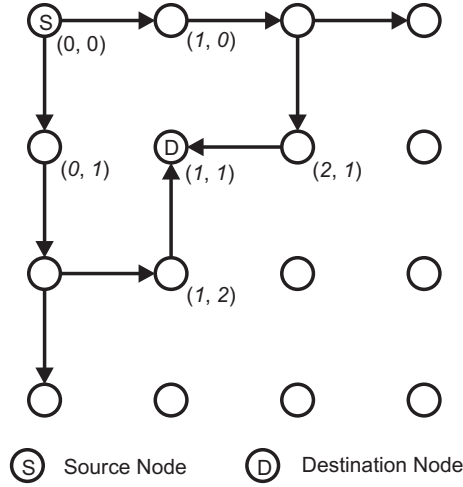


Figure 3.7 Guaranteed Message Delivery. To avoid confusion, the arrows only denote the route of the messages to the destination instead of the whole flooding routes. Since the links from (0, 1) and (1, 0) are unavailable, messages are flooded to (1, 1) via (1, 2) or (2, 1).

As we can see in Figure 3.6, because messages are forwarded by multicast instead of broadcast, even if we would not implement any other algorithms to make further improvement, the redundancy produced by the partial flooding has already been reduced to a quarter of the full flooding in AODV path discovery process.

As in ZigBee routing, redundant identical messages shall be dropped. If a message with the same ID and source address has been received before, the link cost field shall be compared. The one with lower link cost shall stay in the route discovery table, the other one shall be discarded.

The partial flooding shall be optimized further to deal with various cases when a destination and a source have a closer distance in the direction of X or Y or even coaxial, which is going to be discussed in the last chapter.

3.3.3 Hybrid Adaptive Flooding

In our scenario, the partial flooding floods only a quarter area as a full flooding, which means redundancy and energy consumption are also reduced to a quarter; however, in a WSN with a steady wireless channel, the partial flooding still appears excessive; in addition, in the case that a source and a destination are respectively located on the opposite corners of a network, the partial flooding does not differ from a full flooding. In this section, we are going to discuss a hybrid approach which implements the partial flooding with the cooperation of the shortest-path routing to reduce routing cost further. Whereafter an adjustable parameter of flooding hops is used to control the partial flooding and make it adaptive.

For the shortest-path routing, a single hop failure means that the whole path fails. In a WSN with a single-hop failure probability p , the probability of path failure P over n hops can be easily obtained as below.

$$P = 1 - (1 - p)^n \quad (3.2)$$

It is clear that, to reduce the P , either messages are delivered over less hops, or multiple individual routes are provided accessible to the destination. The probability of a successful delivery via m uncoupling paths thus to be

$$P_s = 1 - P^m = 1 - [1 - (1 - p)^n]^m \quad (3.3)$$

In the path discovery phase of the hybrid flooding, route request messages are flooded to several neighbor nodes from where they subsequently discover their own way to the destination. The hybrid routing not only provides multiple paths but also reduces redundancy of flooding.

To have a control on the flooding, we define a key parameter in the route request frame:

- **Definition 3.7:** *Flooding Counter K* - K is configured at a source node as an overhead parameter of a route request command frame to control the range of flooding. K decreases by 1 after each hop, and counts down to 0 after K -hop, and then flooding is consequently over.
- **Definition 3.8:** *Tier K* - At the end of the K -hop partial flooding, route request messages reach an array of $K+1$ nodes, which is called *Tier K* , as shown in Figure 3.8. All $K+1$ nodes on the *Tier K* have the following common property.

$$|X_K - x_s| + |Y_K - y_s| = K \quad (3.4)$$

Where (x_s, y_s) is the source address, and (X_K, Y_K) is the address of a node on the *Tier K* .

For the flooding counter K is applied to the partial flooding, the proposed *Address-Based Adaptive Flooding (ABAF)* routing protocol is accomplished. In the *ABAF* routing protocol, route requests are flooded for K hops to the tier K , and then routed by the shortest-path routing from $K+1$ nodes at the edge of the flooding area to the destination, see Figure 3.8. The redundancy involved in the path discovery is controllable so as to adapt the wireless environment. As mentioned in the previous section, following the partial flooding, three optional shortest-path strategies are available. They show different properties while a common destination is accessed via multiple paths in the systematic topology.

- **X-Y routing** The load obviously gathers on the edge of the topology.
- **Random Walk** Two paths originating from two adjoining nodes on the tier K have a 0.25 probability of overlapping each other on the first hop.

- **Zigzag** Compared with the other two routings, it provides the minimum number of overlapped hops, which ensures the diversity of the multi-paths.

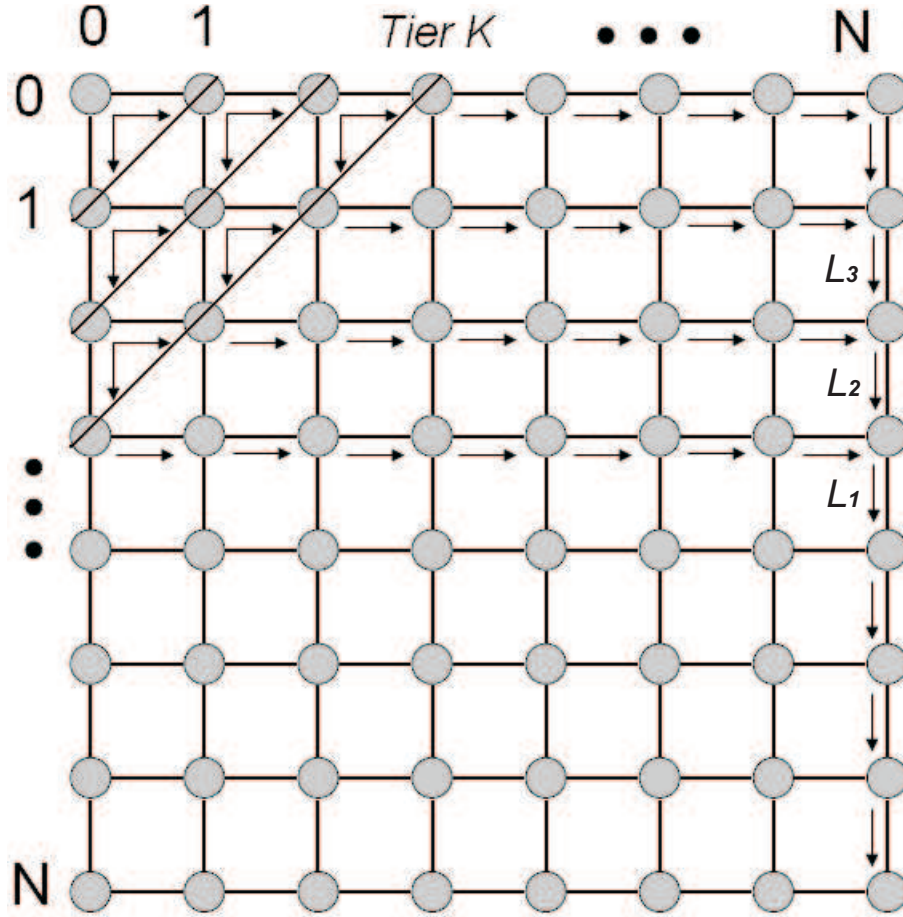


Figure 3.8 The Route Discovery Process of The ABAF Routing Protocol in a regular $(N+1) \times (N+1)$ grid

For what is supposed to be investigated is how the partial flooding and the shortest-path routing interact, it actually does not matter which shortest-path method is employed after the partial flooding. Among the three shortest-path strategies, Zigzag strategy creates routes with the least overlaps and the highest robustness, but it is the most difficult to be generalized to a mathematical model; whereas, X-Y routing produces simpler and more regular paths for analysis, as shown in Figure 3.8; hence, X-Y routing is used for the analysis of the successful message delivery ratio. The following proposition is made.

Proposition 3.2: For a single path, one hop failure means that the whole path is

disconnected. Since X-Y routing gathers the load on the edge of the section, one link failure on the edge will break all the routes over it, while having no impact on the routes behind it. This concludes that each different link failure circumstance on the edge shall be considered and summed up to calculate the successful delivery ratio, i.e., in Figure 3.8, the failures of paths over L_1 , L_2 and L_3 shall all be counted.

In addition, due to the inherent redundancies in flooding, a reasonable assumption is made here.

- **Assumption 4:** In the partial flooding stage of the path discovery process, link failures are not considered.

Consequently, to deduce the successful delivery ratio of a route request message, only the multiple shortest paths between the tier K and the destination (N, N) need to be considered.

An example is given here, both to prove the proposition 3.2 and to explain the following derivation.

Example: In an given 8×8 *ABAF* network model with the link failure probability p , the source node $(0, 0)$ needs to communicate with the destination node $(7, 7)$. While the route request message is transmitted for different distance by the partial flooding, what is the corresponding successful delivery ratio P_s ?

Derivation:

The different distance that the route request is flooded for means the different value of the flooding counter K ; therefore, the corresponding successful delivery ratio can be expressed as the following formulas of the link failure probability p .

When $K=0$, *ABAF* becomes a shortest-path routing. The whole path would be built

successfully only if each single hop was successful.

$$P_s = (1 - p)^{14} \quad (3.5)$$

When $K=1$, the path is actually a **Y** pattern. The whole path would failed if the overlapping links failed or both of the non-overlapping parts failed; consequently, the formula shall be

$$P_s = 1 - \{[1 - (1 - p)^6] + (1 - p)^6[1 - (1 - p)^7][1 - (1 - p)^7]\} \quad (3.6)$$

When $K=2$, referring Figure 3.8, route request messages are unicasted from three intermediate nodes. On the edge, both the failure of the path over L_2 and the failure of the path over L_3 shall be calculated.

$$\begin{aligned} P_s = & 1 - \{[1 - (1 - p)^5] + (1 - p)^5 p[1 - (1 - p)^7] \\ & + (1 - p)^6[1 - (1 - p)^7][1 - (1 - p)^6][1 - (1 - p)^6]\} \end{aligned} \quad (3.7)$$

When $K=3$,

$$\begin{aligned}
P_s = & 1 - \{[1 - (1 - p)^4] + (1 - p)^4 p [1 - (1 - p)^7] \\
& + (1 - p)^5 p [1 - (1 - p)^7] [1 - (1 - p)^6] \\
& + (1 - p)^6 p [1 - (1 - p)^5] [1 - (1 - p)^7] p [1 - (1 - p)^6] [1 - (1 - p)^5]\} \quad (3.8)
\end{aligned}$$

The derivation finally leads to *the general function* of the probability that at least one route from the source node $(0, 0)$ to the destination node (N, N) is successfully built by the *ABAF* routing protocol.

When $K=1$

$$P_s = 1 - \{[1 - (1 - p)^{n-1}] + (1 - p)^{n-1} [1 - (1 - p)^n] [1 - (1 - p)^n]\} \quad (3.9)$$

When $2 \leq K \leq n-1$

$$\begin{aligned}
P_s = & 1 - \{[1 - (1 - p)^{n-K}] + p \sum_{x=0}^{K-2} [(1 - p)^{n-K+x} \prod_{y=0}^x [1 - (1 - p)^{n-y}]] \\
& + (1 - p)^{n-1} [1 - (1 - p)^{n-K+1}] \prod_{z=0}^{K-1} [1 - (1 - p)^{n-z}]\} \quad (3.10)
\end{aligned}$$

Here n is both the horizontal distance and the vertical distance from $(0, 0)$ to (N, N) : in the $(N+1) \times (N+1)$ network model, they are same in hops. The x , y and z are dummy parameters which are produced while generalizing the equation. For this example, the partial flooding is divergent before the diagonal and convergent after

crossing it; therefore, it is meaningless for the equation derivation to set the flooding counter K a value over $n-1$, but a larger value does increase the successful delivery ratio in practice; in addition, in a scenario without network boundary, a $K n$ can make shortest pathes access a destination via all four of its available links, which has been discussed previously.

For the cases that the rectangular section between a source and a destination belongs to a larger network as shown in Figure 3.4, if the partial flooding crosses the diagonal, the succedent shortest-path routing can make route request messages access the destination via all four incoming links, as shown in Figure 3.7. The analysis of this situation would be done in the future work along with another situation that the source node and the destination node are *coaxial*.

- **Definition 3.9:** *Coaxial Nodes* - Nodes in a n -dimensional hypercube, which have $n - 1$ same address components, are called Coaxial Nodes.

The partial flooding crossing the diagonal obviously produces more redundancy as the increase of K ; therefore, it should have a positive impact on the successful delivery ratio theoretically. For the coaxial case, the basic idea is still to multicast towards the destination at the flooding stage; however, the difference is that the multicast section is no longer a quarter but a half of of a full flooding.

3.3.4 The Route Discovery Operation of *ABAF*

Figure 3.8 shows the process that a route request message is oriented from the source node $(0, 0)$ to the destination node (N, N) by the *ABAF* routing scheme, where the partial flooding is followed by the X-Y shortest-path routing. $K+1$ paths leading to a destination are finally generated, and then merge gradually due to the nature of the orthogonal grid. Accordingly, the route discovery process of the *ABAF* protocol operates as described below.

Initiation of Route Discovery

The route discovery (see Figure 3.9) shall be initiated by the NWK layer on receipt of a DATA message generated by the APP layer, for which there is no routing table entry corresponding to the destination address. The DATA shall be buffered, and then a *route request* (RREQ) message shall be generated to initiate the path discovery. The NWK layer shall compare the corresponding address components of the destination (N , N) with its own address (0 , 0). The comparison leads to the multicast address of the partial flooding ($0+$, $0+$) which shall be set in the routing header afterwards.

The flooding counter K shall be set in the corresponding overhead field, either manually following particular requirements or automatically according to the statistic like link quality or data delivery ratio. If the K is not set as 0, it shall be decreased by 1 and saved before the RREQ frame is passed to lower layers. The following important fields shall be contained in the RREQ frame:

- *Source Device Address*
- *Destination Device Address*
- *Multicast Address (Next-hop Address)*
- *Flooding Counter K*
- *Route Request ID*
- *Source Sequence Number*
- *Destination Sequence Number*
- *Hop Counter*

A corresponding routing table entry shall be established and set to DISCOVERY_UNDERWAY as in AODV, and a corresponding route discovery table entry shall be created as well. If the corresponding routing table entry exists, but the status is not ACTIVE, then the entry shall be set to DISCOVERY_UNDERWAY as well.

A route request counter shall be set and maintained to identify the RREQ along with the source device address. An intermediate node shall drop redundant request frames with the same ID. Meanwhile, each route request item in routing table is set an expire timer. The expired request entries with the status of DISCOVERY_UNDERWAY shall be deleted. Once a route discovery request entry is created, the RREQ command frame shall be passed to the MAC sub-layer for multicast. A route discovery request is allowed to be retried by the NWK layer for *nwkInitialRREQRetries* times as defined in AODV.

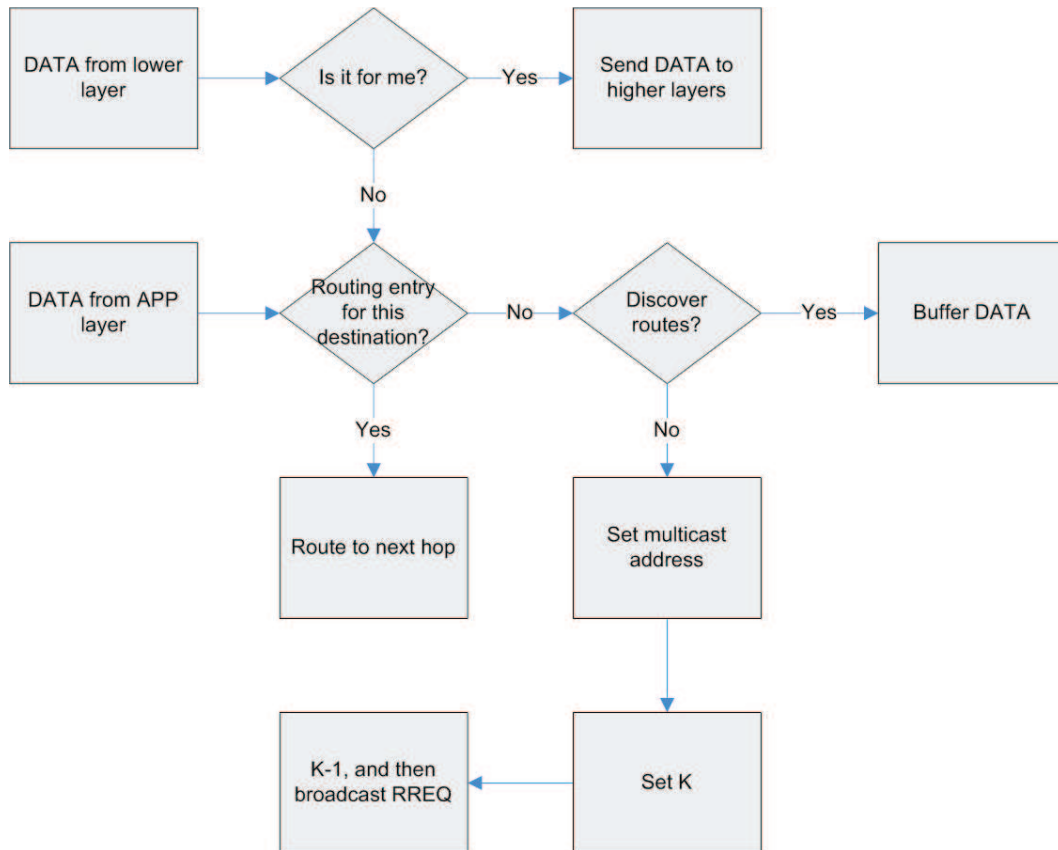


Figure 3.9 Basic ABAF Route Discovery Algorithm

On Receipt of a Route Request

In the partial flooding stage, when an intermediate node is upon receipt of a RREQ frame, it shall firstly check if itself is in the multicast section of the RREQ, if not, it shall discard the RREQ. If the node is in the multicast section, it shall check if there is a

route discovery table entry with the same ID and source address existing. The purpose of this step is to discard redundant messages and avoid loop. If a route discovery table entry with the same ID and source address exists, the respective link cost shall be compared, the one with lower link cost shall be in the route discovery table. If no such entry exists then one shall be created, and set an AODV *nwkRouteDiscoveryTime* milliseconds expire timer. The next step is to check the corresponding routing table entry with the same destination address. If the routing table entry is not ACTIVE or does not exist, one shall be created and set to DISCOVERY_UNDERWAY. Following the routing table entry, the flooding counter K shall be checked. If K is not zero, it shall be decrease by 1, and the RREQ shall be rebroadcast; otherwise, a zero value K means the end of flooding. The multicast address shall be replaced by an unicast next hop address that shall be produced by the shortest-path routing stipulated in the previous section. The RREQ frame is then passed to MAC layer for unicast.

In the shortest-path stage, as the address field in the route request frame overhead has been placed by an unicast address, an intermediate node shall firstly compare the unicast address with its own, and then make a choice to process it or drop it accordingly. The subsequent procedure shall follow the same as in the flooding stage except the flooding counter K that does not need to be processed as it has reached zero.

If either the node is the intended destination or the corresponding routing table entry with a greater destination sequence number than RREQ frame is available, a route reply command frame shall be generated and replied along the reverse path set up by RREQ frame. Figure 3.10 illustrates how to process route request both in the partial flooding stage and the shortest-path stage.

3.3.5 Tentative Scheme on Route Maintenance

The systematic address structure and the ABAF routing do require and also consequently cause some modifications on the route maintenance strategies. As a part of our

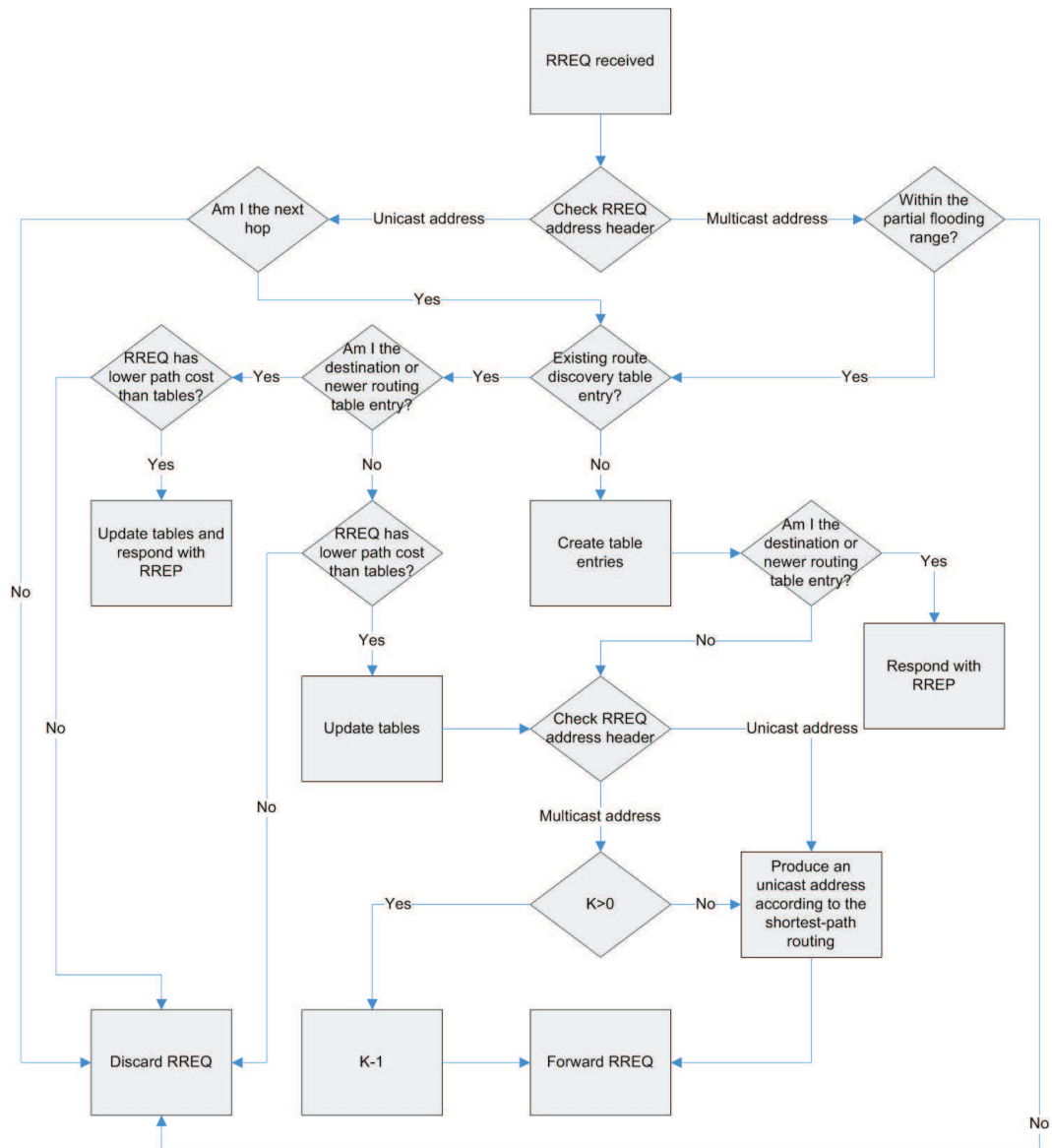


Figure 3.10 Receipt of Route Request in ABAF

work, they are discussed in this section; however, they are not proven and simulated in this dissertation, probably covered in the future work.

In ZigBee routing protocol, a neighbor table contains the information of every device within the transmission range. The field of relationship and device type are used in ZigBee tree or cluster structure to denote the status of nodes. In the scenario of *ABAF*, the routing focuses on end to end communications, and all nodes are peers in the systematic address space regardless of the hierarchical mode of address assignment; therefore, the field of relationship and device type are not required in this case. In addition, it is suggested that only the four one-hop neighbors are maintained in detail, while a brief record of all other neighbors within the transmission range is needed, in practice, to cope with node failures. The methods above are supposed to reduce the size of neighbor tables effectively.

Since all messages are sent out via the four outgoing links, the routing table entries can be categorized into an index of these four links so as to reduce the routing table size and increase the searching speed of the entries.

When a link failure occurs, the node upstream of the failed link shall report the link failure to the source node in the same way as the ZigBee routing. A little modification is suggested to be done on the *local repair* mechanism of AODV: when a link failure occurs in an active route, instead of broadcasting a RREQ directly to rediscovery a route in AODV, the node upstream of the broken link shall firstly identify the multicast section of the destinations that are effected by the failed link, and then multicast a RREQ to the target section via the other available link. If both of the links towards the destination is broken, a broadcasting shall be generated.

The key information that are suggested to be included in the neighbor table and the route table are listed in tables below.

Table 3.3 enumerates the values for the route entries.

The description of the *ABAF* routing protocol in this dissertation is a proposed draft

Table 3.1 Neighbor Table Entry

Field Name	Description
Network Address	The network address of the neighboring node, which is an address assigned by the addressing scheme when nodes join in the network.
Link State Information	The field used to estimate link quality, which may include Link Quality Indicator (LQI), path cost, transmit failure counter, <i>Hello</i> message counter and beacon counter.

Table 3.2 Routing Table

Field Name	Description
The Link $\leftrightarrow (x+1, y)$	Entries of the routes via the link. See Table 3.3.
The Link $\leftrightarrow (x-1, y)$	
The Link $\leftrightarrow (x, y+1)$	
The Link $\leftrightarrow (x, y-1)$	

Table 3.3 Routing Table Entry for The Four Valid Links

Field Name	Description
Destination Address	The network address of the destination of this route entry.
Status	The status of the route. Refer to [3].
Multicast Section	The field shall be the multicast address when the routing process is in the partial flooding stage, and empty while the shortest-path routing.
Next-hop Address	The field shall be the network address of the next-hop node on the way to the destination when the shortest-path routing is functioning; otherwise, it shall be empty.

work that shall be regulated and standardized in a formal specification; however, it is out of scope of this thesis.

Chapter 4

SIMULATION AND RESULTS ANALYSIS OF ABAF

In the experiments described in this chapter, *ABAF* is simulated using an object-oriented modular discrete event network simulator called OMNeT++ [28], in which a simulation model consists of hierarchically nested modules programmed in C++. The objective of this experiment is to practise the *ABAF* routing protocol and prove its feasibility in a given dynamic environment; furthermore, the performance of *ABAF* shall be compared with ZigBee routing protocol (a simplified AODV) to show that the path discovery stage of *ABAF* produces less redundant transmission while achieving a satisfied successful delivery ratio.

4.1 SIMULATOR

Several simulators are available for simulation and testing of protocols and algorithms.

- *Ns-2* is probably the most popular and network simulator; however, it does need advanced skills to perform a meaningful simulation. It focuses on the ISO/OSI model simulation.
- *GloMoSim* stands for Global Mobile Information Systems Simulation Library that currently provides only protocols for the simulation of pure wireless networks.
- *SENSE* is a specific simulator for the simulation of sensor networks. The simulation capacity depends on the communication pattern of the networks, which could drop to an upper limit of 500 nodes.

- *TOSSIM* is the simulator for TinyOS motes and thus to be platform-specific. It compiles TinyOS codes to executable files that can be directly run on a PC.
- *Shawn* is a new simulator developed for the simulation of large scale networks, which currently needs to be further developed, and provide more technical supports and contributed models.

A comparative chart of some popular simulation tools is given in [29] regarding the criteria of the abstraction level and scalability, as shown in Figure 4.1, which is summarized by the author: *"This does not express the maximal feasible network sizes, but rather reflects the typical application domain."*

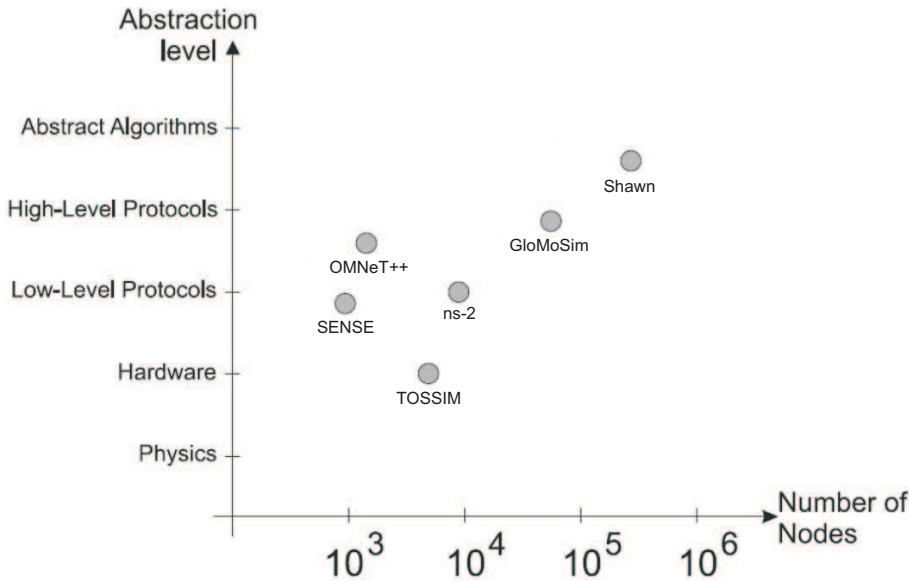


Figure 4.1 Simulators for Different Application Domains

OMNeT++ is a frequently used simulator for WSNs, can run on both Microsoft Windows and Unix OS using various C++ compilers. The modular simulation model allows users to reflect the logical structure of an actual system in the flexible hierarchical modular structure. Various module parameters allow users to control every module using either the command line interface or the graphical user interface (GUI). The simulator also provides abundant base function modules called simple modules that are programmed in C++ using the simulation library. An advanced GUI makes the inside of modules and the process of simulation execution visible and controllable by users.

The following components are the basic elements to run an OMNeT++ simulation.

- The configuration file, usually called *omnetpp.ini*, is firstly processed by the program when a simulation is run. This file contains the settings needed for the execution of a simulation, values of model parameters, etc..
- Module and topology properties are described in *.ned* files, which are parameters, gates etc..
- Message definitions are user-defined and will be translated into C++ classes.
- Simple modules sources are programmed in C++ files, which are the functioning modules of a simulation.

The decision to select OMNeT++ as the simulator in the present work is supported by the following considerations.

- **Object-oriented Language C++** C++ is probably the most popular object-oriented language that just caters to the concept of modular simulation; in addition, C++ is one of the fundamental computer languages, easy to start and use.
- **Modular Simulation Structure** Modular structure is highly flexible: modules from different model packages can be modified and combined to function freely, as well as your own modules.
- **Excellent Technical Support** There are abundant models on OMNeT++ website contributed by the users all over the world. The forum and the news group of OMNeT++ provide effective interaction, help and support. The AODV model used in this work, Ad hoc Sim [30], is a contributed model that can be downloaded from OMNeT++ website.
- **Free Software** OMNeT++ is currently free for academic and non-profit use. The commercially supported licensed version, OMNEST, can be purchased from Omnest Global, Inc.

4.2 AD HOC SIM

Ad hoc Sim [30] is a contributed OMNeT++ model for ad hoc networks, in which AODV protocol and several mobility models are implemented. It provides a simple platform on which the simulation model of *ABAF* is developed; therefore, the description of the *ABAF* model starts from introducing Ad hoc Sim. The simulator is able to build an ad hoc network with a user-defined number of hosts that move in a free 2-dimensional field, as shown in Figure 4.2. The main features include:

- A single mobile node is a encapsulated compound model composed of the basic OSI modules.
- Six mobility algorithms are provided as the mobility module. Each node can move following an independent algorithm.
- Links are dynamically built and disposed.
- Both unidirectional and bi-directional links can be set up between adjacent nodes depending on the respective transmission power of each node.

The following modules are encapsulated in each node, also see Figure 4.2:

- A physical layer.
- A MAC layer.
- A route layer.
- An application layer.
- A mobility layer.

The routing module is based on AODV standard draft v.10. The modules communicate through messages exchange. Both single modules and compound modules can be embedded, replaced or regrouped by simply modifying the *omnetpp.ini* file. Statistical

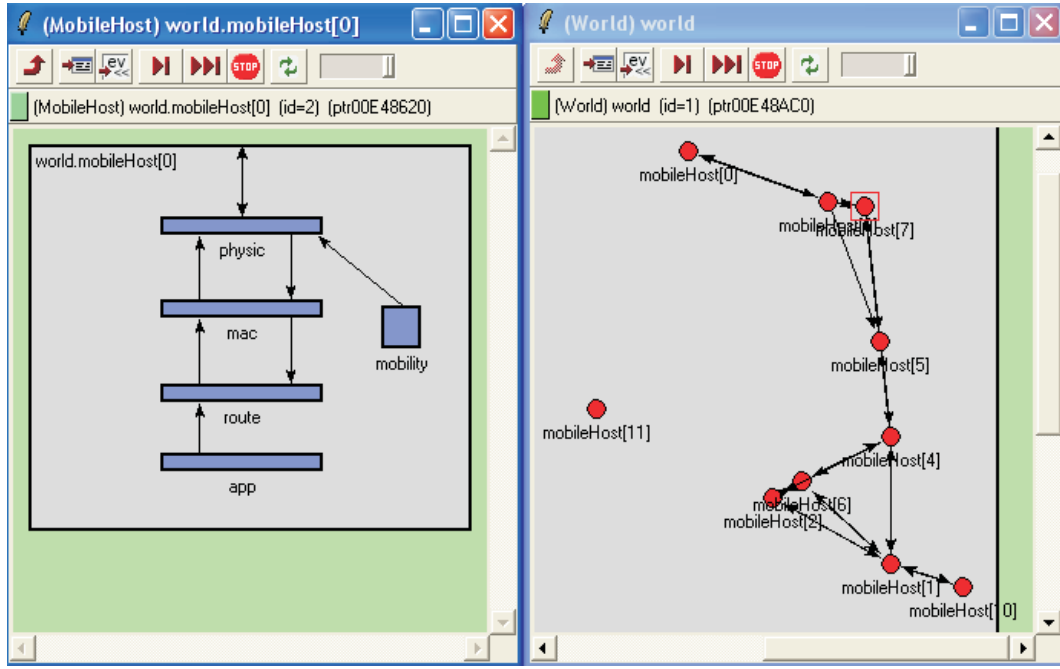


Figure 4.2 The Simulation Topology and The Host Internal Structure of Ad hoc Sim

data is recorded in text files for further analysis at the end of the simulation. A bug in the kernel of 2.2 version OMNeT++ rewrites the statistical data file on every new execution; therefore, a script file is advised by the author to backup the collected data automatically at the end of each simulating. The newly released version 3.0 of OMNeT++ has fixed this bug. Because the text file produced by Ad hoc Sim can not be read by the OMNeT++ analysis tools, the output method of the *ABAF* model was modified using the default OMNeT++ classes.

The detailed description of each OSI layer module is covered in the rest of this section.

4.2.1 Mobility Modules

The only Ad hoc Sim module, which does not belong to the OSI structure, is the mobility module. How a node moves from a position to another is determined by the mobility module currently used in the simulation. The mobility module implements one of the following algorithms to calculate the next step of a mobile node. In a simulation, to assign one of the algorithms just needs to change a string value in the *omnetpp.ini*

file.

- **Random Walk Mobility Module** - The random Walk is the discrete interpretation of the Brownian mobility model, which has a completely unpredictable motion pattern with uncorrelated speed and direction.
- **Restricted Random Walk Mobility Module** - The restricted random walk mobility model tries to reduce the randomness of a movement by choosing the new values of speed and direction angle within a limited range around the former ones.
- **Random Way-point Mobility Module** - The random way-point model has a quite realistic mobility pattern that allows a host model to choose a destination point. The movement towards the destination point is defined by a speed, which is uniformly distributed between the parameters (*minSpeed*, *maxSpeed*). Once the node has reached the chosen point it stays there for a given time called pause time, and then chooses a new destination.
- **Random Direction Mobility Module** - The random direction model is a mix of the random way-point and the random walk models. Each host node chooses a speed and a direction distributed uniformly, and goes on until it reaches the borders. Host nodes also wait a pause time before choosing a new direction and speed.
- **Normal Markovian Mobility Module** - The generation of parameters follows the Markovian model.

Ad hoc Sim also provides two optional behaviors when a node moves to the map borders.

- **Rebounding:** Nodes will be rebounded to the direction of reflection.
- **Toroidal Movement:** Nodes will roam out of the map from one side and re-enter from the opposite side.

The mobility module works as below:

- 1) A self-message shall be scheduled to initiate the mobility module.
- 2) After the self-message is processed, a new position (x, y) shall be generated by the implemented algorithm according to the current position of the node.
- 3) The value of the new position shall be stored in the newly created MOVE message, which will be sent to the physical module as well as used to re-display the node icon in the new position in the graphical topology map.
- 4) As soon as the MOVE message is sent out, another self-message shall be rescheduled to be processed in a *moveInterval*.

4.2.2 PHY Layer

The physical module of each host model is responsible for the dynamic creation of "in" and "out" gates that are used for the messages exchange among the neighbors, see Figure 4.3.

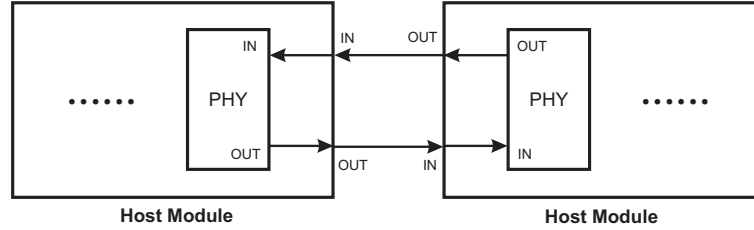


Figure 4.3 The Connection between PHY Modules of Two Nodes

When a node is assigned a new position, which means the node moved, the distance between adjacent nodes shall be calculated. If a new neighbor moves in the transmission range of a node, the following actions shall be taken:

- 1) A new gate shall be created for both the compound modules (the two nodes);
- 2) Within both of the compound modules, a new gate shall be created on the physical module;
- 3) A link shall be created to connect the new gate on the physical module and the new gate on the host compound module;

- 4) A link shall be created to connect the two compound modules.

When the distance to a neighbor node is out of the transmission range after a movement, the corresponding gates shall be deleted, as well as the links that connect them.

The physical module mainly deals with messages from three sources, neighbors, the mobility module and the MAC layer. When a message from a neighbor node arrives, it shall be sent directly to the higher levels if no errors. The messages from the mobility module, the new position value, shall be stored and used to update the position of the host module. When a message is passed from a higher level module, the physical module shall scan the current neighbor list for the "out" gates, and then send a new copy of the original message through the gates. The simulator kernel records the delivery time of the message to the neighbors according to the gate settings and the message length.

4.2.3 MAC Layer

Ad hoc Sim implements a simple MAC layer module instead of the commonly used CSMA/CA. The outgoing messages are simply passed to the PHY layer module, while the incoming messages from PHY module are processed by a *MM1 queue* policy. The higher level state shall be checked on receipt of an incoming message from PHY module. If the state is busy, the message shall be buffered, or dropped if the buffer is full; otherwise, the incoming message or the first message in the buffer shall be sent to the higher level. An *end of service* message shall be scheduled afterwards to trigger a new pick from the buffer or set the state of the higher level.

The default mode of the MAC layer module, *promiscue mode*, allows any messages from outside to be passed to the upper layer for convenience; however, Ad hoc Sim also optionally enables its MAC layer module to check the mac address of an incoming message and pass those who are broadcasted or addressed to this node.

4.2.4 AODV Routing Module

The routing module is the core of Ad hoc Sim. All the AODV control messages are set 512 bytes long, and all the control fields are attached in the control messages as parameters that can be configured either by the GUI on real time or by programming the module codes.

The AODV standard provides many optional operations to make the protocol as adaptable and flexible as possible; however, implementing all of the operations is complicated and unnecessary. Ad hoc Sim implements the basic options suitable for an ad hoc network:

- **HELLO Message** HELLO messages are exchanged between neighbors to check the status. In an Ad hoc Sim simulation, a self HELLO message is scheduled to initiate the simulation.
- **Expanding Ring Search Optimization** This operation initiates a route request with a small *tll* hoping to obtain a route from some neighbors. If no route reply is received another request with a bigger *tll* shall be sent. A fixed number of retrials is allowed, after which the transmission trial shall be aborted.
- **ACK Message** In Ad hoc Sim, ACK messages are used to confirm the correct delivery of both a RREP message and a DATA message, while AODV standard uses it only for a RREP confirmation.
- **Blacklist** A blacklist is used to block unreliable neighbor nodes. As each node has different transmission range, a node A can hear a node B while the node B might not be able to detect the node A. In this situation, The node A shall insert the node B in the blacklist if it can not receive any acknowledgment after trying a number of times to send a RREP message to A. The link between A and B thus to be unidirectional. In AODV standard, the blacklist only block new RREQ messages passed from B for a fixed amount of time, while Hello messages are still processed by A.

Ad hoc Sim defines AODV messages with a constant value and various parameters. Self messages are used as interval timers for events like a route expiration or a route request time out. AODV messages employed in the routing module are listed as below:

- *HELLO*: A Hello message;
- *RREQ*: A Route Request message;
- *RREP*: A Route Reply message;
- *RERR*: A Route Error. It contains a list of invalid destinations;
- *DATA*: A Data message;
- *RREP_ACK*: A RREP Acknowledgment message;
- *DELETE*: A self message scheduled to trigger a route expiration event that deletes an invalid route. According to AODV standard, a *DELETE* messages shall be self-scheduled after the previous one is executed;
- *FLUSH*: This message handles the RREP time out. All the buffered *DATA* messages waiting for the RREP shall be deleted after a fixed number of time outs;
- *SEND_HELLO*: A self message to trigger a new Hello message;
- *BLK_LIST*: A self message to put a neighbor in the blacklist.

The messages above are only a group of carriers, the control information is actually denoted by the attached parameters. The most important and commonly used parameters in AODV messages are listed below:

- *originator*: It is the ID of the node who generates the *RREQ*. It will be copied in the corresponding *RREP* message to orient the *RREP* back to the originator;
- *dest*: It is the ID of the node that the originator wants to communicate with, only used in *RREQ* messages;

- *seqNumS*: It denotes the sequence number of a message generated by the originator. This field will remain the same;
- *seqNumD*: It denotes a sequence number that is related to the last known destination of the message. The value 0 means no destination was known;
- *source*: It is the ID of the last upstream node, from where the message comes from;
- *mac*: It is the ID of the next hop of the message;
- *tll*: The lifespan of messages. It is denoted by number of hops;
- *hopNum*: The hop counter that records how far the message has traveled so far.

When the finish function of an Ad hoc Sim simulation is called, the following information shall be collected for analysis.

- The number of hops that a message traveled is stored in a *histogram array*, which can be inspected at run-time;
- The latency of a message delivery is recorded, and the latency mean is calculated according to the number of hops;
- The throughput is calculated by the message size over the latency;
- A delivered message counter is maintained for the successful delivery ratio.

4.2.5 APP Layer

All the DATA messages in a simulation are generated by the application layer module and passed to the routing module to trigger the routing operations. Each node generates the data traffic independently, which can be switched on or off by setting the corresponding *app.active* parameter in the *omnetpp.ini* file. In the same way as the other modules, the initiation of the APP module is triggered by a self message. The

rate of sending messages is defined by the rate parameter, while the interval between two bursts is defined by *burstInterval* parameter. The data traffic is modeled as a burst of sixty four messages, which can be defined by the parameter. The destination is randomly chosen and stays the same for the whole burst of DATA messages. Each host module is identified by an ID number that is assigned by the OMNeT++ kernel at the beginning of simulation. These IDs are generated in a certain sequence and stored in a table of all host modules in the simulation, which can be called by any other modules through pointers.

4.3 SIMULATING AODV IN A STATIC NETWORK

Ad hoc Sim implements AODV routing in a mobile manner by mobility modules. It is easy to simulate AODV in a static network by modifying Ad hoc Sim. To differentiate from Ad hoc Sim, the modified version is temporarily called Fixed Ad hoc Sim (FAS).

4.3.1 Fixed Topology

The following main steps are done to turn the mobile network model of Ad hoc Sim into a set of fixed host modules.

- 1) Remove the mobility module;
- 2) Assign an address to each generated nodes;
- 3) Associate the address of a node with its host module number that is allocated by the simulator;
- 4) Position the nodes on the simulation map.

The first thing needs to be done is the removing of the mobility module, which is companied by disabling the corresponding option in the *omnetpp.ini* file and deleting the corresponding gates in the PHY module. In the network model of *ABAF*, the address of a node reflects the position at the same time; therefore, to differentiate from the X and Y in Ad hoc Sim, C (column) and R (Row) are used here to denote

the address of a host module as well as the position. However, both the address and the position can not be recognized by the simulator by now, because the host module number is the only "official" identification that is assigned and used to locate the host module by the simulator; therefore, the address has to be associated with the host number. The position of a host module in the simulation map is computed according to the address of the node. Both of the association above are carried out by a class *setPos*, see Table 4.1.

Table 4.1 Algorithm 1: The class *setPos*

```

class Physic : setPos()


---


  hostModNum = the index of the parentModule;
  C = hostModNum % columnNum;
  R = hostModNum / columnNum;
  if hostModNum < columnNum
    posC = hostModNum × the one-hop distance;
    posR = 0;
  else
    posC = C × the one-hop distance;
    posR = R × the one-hop distance;

```

In addition, to achieve the customization of the size of network, two dialog boxes are produced to allow users to input the number of columns and rows of the network at the beginning of the simulation, see Figure 4.4.

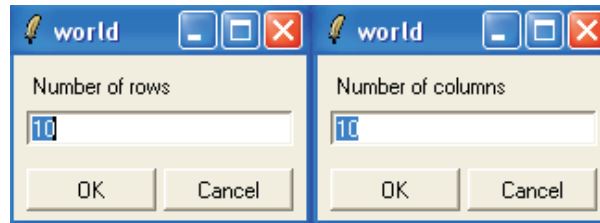


Figure 4.4 The Input Dialog Boxes for The Number of Columns and Rows

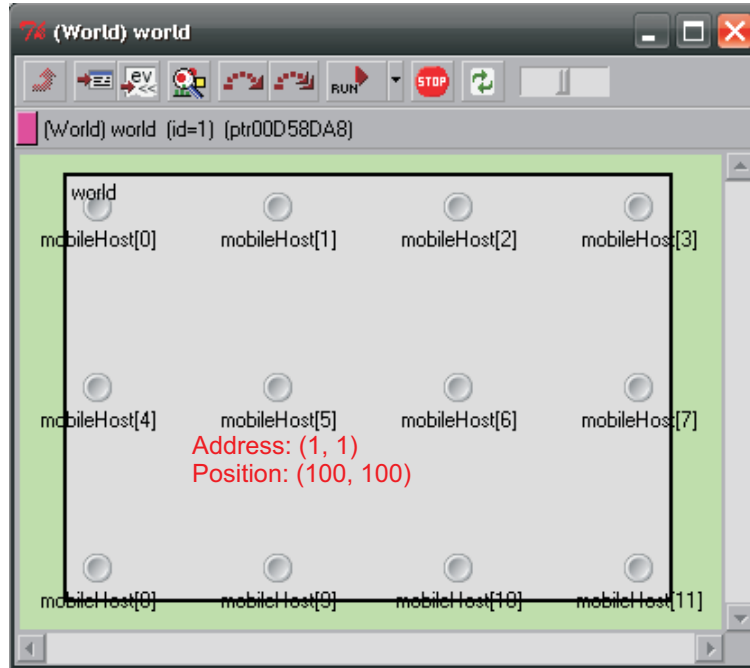
The modification done to the *.ned* files and modules are summarized in Table 4.2 and 4.3, and a network model generated by FAS is shown in Figure 4.5, where the one-hop distance between neighbors is set a value of 100.

Table 4.2 Algorithm 2: The Modified Network Module *world.ned*

The modified network module: world
module World
parameters:
Insert Row and Column ;
submodule: hostModule
parameters:
Insert R and C here; //as well as in <i>hostModule.ned</i>
network World
parameters:
row = input;
column = input;

Table 4.3 Algorithm 3: The Modified PHY Module

The modified simple module: physic
initialize()
define C and R ; //was a random X and Y
setPos();
//was a class used to display the hostModule on the initial position (X, Y)
handleMessage()
no more messages from Mobility module are handled;

**Figure 4.5** A 4×3 Static model Generated by FAS

4.3.2 Transmitter Module

The transmitter module is the most important module excepting the routing module, for it is the module that simulates the dynamic wireless environment. The signal degradation follows the *Free Space Propagation* model, see Equation 4.1, which states that the received signal strength is inversely proportional to the square of the distance.

$$P_r = \frac{P_t G_t G_r \lambda^2}{(4\pi d)^2 L} \quad (Watts) \quad (4.1)$$

Where

P_r is the receive power;

P_t is the transmit power;

G_r is the gain of the receiving antenna;

G_t is the gain of the transmitting antenna;

λ is the wavelength of the carrier signal;

d is the distance between the receiver and the transmitter;

L is the system loss factor.

In Ad hoc Sim, the transmission power is pre-configured; therefore, the movement of nodes makes the networks dynamic. In reverse, the distance is constant in fixed network, changing the transmission power thus to become the way to simulate a dynamic environment where broken links are caused by interferences. The transmitter module is built to provide the values of the transmission power to the PHY module dynamically, so that the coverage of nodes changes, so does the probability of the link failure. The transmitter module is accomplished after a optimization. The original one worked in the similar way as the mobility module in Ad hoc Sim, shown as below:

- 1) A self-message is scheduled to initiate the module.
- 2) The self-message triggers the generation of a new *transmitMsg* message with a new transmission power within a certain range.
- 3) The new message is sent to the physical module and used to re-calculate the transmission range.
- 4) Right after the *transmitMsg* message is sent out, another self-message is rescheduled to be processed in a *transInterval*.

The results from the simulations using the original transmission module were unusable, which showed that the delivery ratio varied dramatically from one simulation to another. There were often no messages delivered in a simulation. The analysis led to the following reason:

- The parameter *transInterval* acts as a window timer that determines the frequency of rebuilding the links throughout the network; therefore, an appropriate *transInterval* is essential to simulate a expected wireless environment. However, other events in the simulation might not be synchronous as the *transInterval* due to the random uncertainty in the presence of link failures; therefore, it frequently happened that the path becomes unavailable during the transmission of messages.

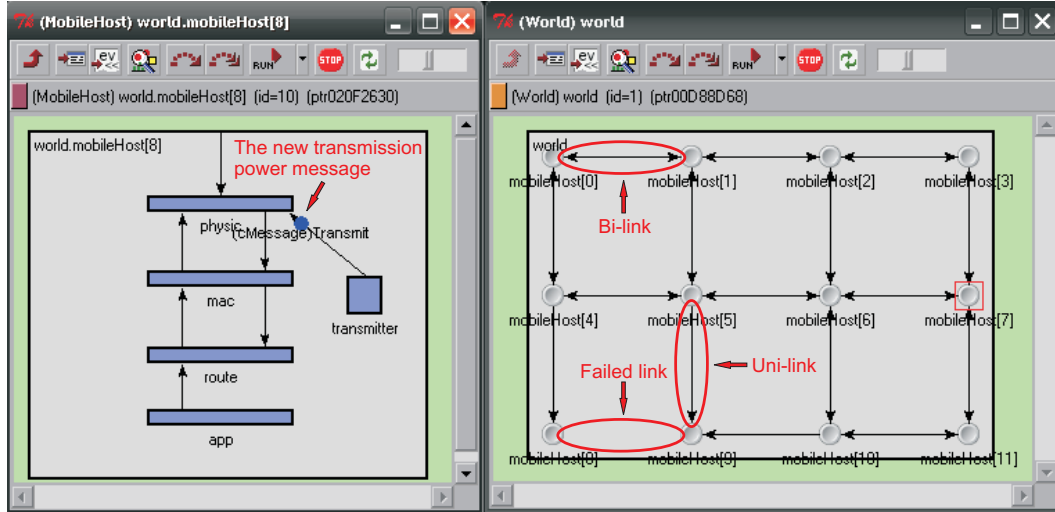
Based on the analysis, the modification was done to the original module. The finalized transmitter module is shown in Table 4.4.

In the new module, any incoming message will reset the timer. Accordingly, a notification message is defined in the PHY module, which is sent to the transmitter module each time a DATA message is successfully sent out by the PHY module. Consequently, a sent DATA message by the PHY module will trigger a rebuilding of the outgoing links, which will provide a new link with the link failure probability p for the next DATA message, and then, the *transInterval* timer is reset. This strategy not only prevents the data delivery from the unexpected interruption by a link rebuilding, but also ensures a link rebuilding in case of a dead circle that neither outgoing links are available nor DATA messages are generated.

Table 4.4 Algorithm 4: The Transmitter Module

Simple module: Transmitter
<pre> initialize() initiate the <i>transInterval</i>; schedule a self message; initiate statistical variables; handleMessage() handle the self message and generate a new transmission power; generate a new message; send the new transmission power message to the PHY module; if a self message is scheduled //reset the timer cancel the self message; schedule a new self message according to the <i>transInterval</i>; </pre>

Each node has its own transmission power changing within a certain range, so links between neighbors can be both unidirectional and bi-directional; therefore, it happens that a node has a link outward but there is not a reverse link, see Figure 4.6 for the unidirectional and bi-directional links generated by the transmitter module.

**Figure 4.6** The Topology with Dynamic Links Generated by The Transmitter Module

The other modifications to make the transmitter module function include registering the module and the parameters in the *omnetpp.ini* file, and defining the module, the parameters, the gates and the connection to the PHY module in the *simple.ned* and the *hostModule.ned* file. Table 4.5 shows how the PHY module interact with the transmitter

module.

Table 4.5 Algorithm 5: PHY Handles a Message from Transmitter

class Physic: handleMessage()
if a message arrived from Transmitter
read the new transmission power value carried by the message;
calculate the new transmission range;
updateConnections();
//update all connections based on the new transmission range delete
the message;
if a message arrived from MAC
updateConnections();
broadcast it;
if the message is a DATA message
send the transmitter module a notification;

4.3.3 Modifications on AODV Routing Module

The *Expanding Ring Search* optimization in the original AODV module would obviously increase latency and produce more redundancy in the path discovery stage of the simulation; whereafter, the impact on both AODV and *ABAF* becomes uncertain due to the dynamic environment. To improve the accuracy and fairness of the simulation, The *Expanding Ring Search* is disabled in both FAS and the *ABAF* model by setting the initial value of *ttl*, *TTL_START*, equal to the *NET_DIAMETER* that is the maximum hops from one side of the network to the other side. Both of the two parameters are defined in a head file *aodvConstants.h*

4.4 *ABAF* MODEL

The benchmark model, AODV, has been successfully simulated in FAS. The *ABAF* simulation model shall be basically accomplished by replacing the AODV routing module with the *ABAF* routing module.

ABAF can be concisely defined as a multicasting AODV cooperating with a shortest-path routing and operating on a systematic address space; therefore, the flooding phase

of *ABAF* can actually employ AODV algorithm with all the messages and parameters. The following key elements of *ABAF* are added on the AODV routing module.

- **The Shortest-path Routing** As stated in the chapter 3, X-Y routing is used as the shortest-path routing scheme in *ABAF*.
- **The Flooding Counter K** K is set as one of the parameters when a RREQ is generated, and processed by the *handleRREQ()* class later on.
- **Addresses** One of the most important phases in the route discovery of *ABAF* is the addresses comparison. Although the assigned addresses can not be recognized by the simulator, they are necessary for the denotement of the network topology and the positions of host modules. However, host numbers are apparently not always sufficient for any algorithms, thus, a mapping has to be done between addresses and host numbers (see Table 4.1).

The *ABAF* algorithm is embedded in the routing module as shown in Table 4.6.

As proposed in the algorithm above, the key phases are to generate and handle RREQ, all the other phases shall be operated in the same way as AODV after a path is built. For all the host numbers of the neighbors are stored in the neighbor list in PHY module, the class *getNexthop()* is programmed in the PHY module to do the comparison job and determine the next hop of a message, which is shown in Table 4.7. As the traffic in the *ABAF* simulation is constantly from the origin point $(0, 0)$ to (N, N) , the class currently does not consider destinations in any other three multicast sections, which will be programmed in the future work.

The host numbers of the neighbors are stored in number order in the list; furthermore, the host modules are displayed from left to right and from the top down; in addition, only maximum two neighbors are in each multicast direction; therefore, the first neighbor checked in the *getNexthop()* function must be the one on the X-axis if available, and accordingly, messages are sent in X-Y fashion.

Table 4.6 Algorithm 6: ABAF algorithm

Simple module: ABAF

```

initialize()
    insert and initiate floodingCount;
handleMessage()
    if message from APP
        call sendData() to process the message;
        broadcast the returned value; //send it to MAC layer
        delete the message;
    else
        case RREQ:
            call handleRREQ() to process the RREQ;
            broadcast the returned value;
            delete the message;
            break;
sendData()
    if no corresponding route available
        buffer the message;
        generateRREQmsg();
        //set TTL_START large enough to disable the Expanding Ring Search;
        remember the new RREQ;
        return the new RREQ;
generateRREQmsg()
    insert the new parameter floodingCount;
handleRREQ()
    if no corresponding route available
        check floodingCount for the next step;
        if floodingCount is larger than zero
            increment the hopCount;
            decrement the floodingCount;
            return the RREQ for rebroadcasting;
    else
        if I am the next hop of the RREQ
            //forward the RREQ along the single Path
            obtain the address of the next hop;
            //a class physic- >getNexthop() is used here
            add the next hop address on the RREQ and return it;
        else
            drop the RREQ by returning NULL;

```

Table 4.7 Algorithm 7: The class *getNexthop*

```

class Physic: getNexthop()


---


for check the availability of all nodes in the neighbor list
    if a neighbor host number larger than me
        return this neighbor;
    break;

```

In addition, the flooding counter K is also configured as a user-input parameter, in the same method as the number of columns and rows.

4.5 SIMULATION ANALYSIS AND PERFORMANCE EVALUATION

4.5.1 Simulation Analysis and Setup

To test and compare the performance of AODV and *ABAF* in the static network, two equal simulation models have been successfully built by now, which includes OSI modules, the network topology and the wireless channel. The parameters are set up referring to [31, 32, 20, 21, 33] and a conference paper based on this work [34]. In the simulations, a 5×5 grid is considered as a small scale network, while a 20×20 grid should be practical for a comparative large scale, the flooding counter K thus to be 1 to 3, and 1 to 18 respectively. The control messages are 64 bytes, and the size of a data packet is set 512 bytes. As shown in [34], in the network with a link failure probability less than 0.05, even a single path routing can have a high message delivery ratio; meanwhile, a link failure probability higher than 0.20 is harsh for most of routing protocols and considered as a extreme value in the simulations in [31, 32, 20, 21]. Therefore, 0.05 and 0.2 are considered as the low and high link failure probability in this simulation.

As discussed in the section of the transmitter module, the way that the transmitter module rebuild the links can make a great impact on the simulation, so does the setting of DATA burst length, DATA rate, DATA burst interval and *transInterval*, which is a noticeable problem and need to be practiced and optimized in the future work.

Accompanied by the transmitter module, the finalization of the parameters setting above also experienced two stages. In the first stage when the old transmitter module was used, the key parameters were set as in Table 4.8.

Table 4.8 The Old Setting of The Key Parameters

Item	Value
Burst length	64
Send Packet Rate	4/sec
Burst Interval	2 sec
<i>transInterval</i>	17 sec

Based on the table above, the data traffic was simulated as continuous bursts of 64 data packets at a rate of 4 packets per *simulation second*. The cycle of a whole data transmission equals to the data sending time plus the burst interval. The *transInterval* was set slightly shorter than the data transmission cycle to avoid events collision during the simulation. The earlier idea focused on simulating more interferences on data delivery; therefore, the burst interval was set very short compared to the data sending time. This setting were supposed to provide a new wireless environment for each data burst and an interference on each built route; however, as analyzed in the section of the transmitter module, the periodical rebuilding of the outgoing links not only interrupts data delivery, but also make a great negative impact on RREQ delivery. The setting of a data burst of 64 packets was originally to increase the resiliency to against the interferences; nevertheless, a further analysis led to a model fault:

- An intermediate node would drop all buffered DATA packets before it can initiate a *local repair* or send a RREQ back to the source node, if a link failure occurred during data delivery.

The reason is complicated. As we know, all the nodes in a simulation are actually copies generated by the simulator, and have totally same functioning modules. The commands or actions from each node are executed by the simulator in the order of the host number. Events are triggered by another event or a self timer. To solve the

fault, all the event timers and the order of events execution need to be rearranged and rescheduled, which should be another big project by estimate and apparently out of scope of this work; therefore, this set of parameters setting was given up.

Along with the employment of the final transmitter module, a new setting of parameters is carried out, which is summarized in detail in Table 4.9, also refer to [30].

Table 4.9 Simulation Parameters

Network	
Number of Hosts	25, 400
Node Enabled to Send DATA	1
Simulation Time	3600 simulation seconds
PHY Module	
Transmission Power	distributed in [9950, 10950], [9800, 10800] ρ Watt
Receive Threshold	1 ρ Watt
Channel Bandwidth	11 Mb/s (IEEE 802.11a)
Channel Delay	10 μ sec
Channel Error	1 bit on 10^6
MAC Module	
Input Buffer Size	1 Mb
AODV/ABAF Module	
Link Failure Probability p	0.05, 0.20
Control Message Size	64 bytes
HELLO Interval	1 sec
Allowed HELLO Loss	2
Max RREQ retry	3
Flooding Counter K	1~4, 1~19
APP Module	
Destination	(4, 4), (19, 19)
DATA Packet Size	512 bytes
Burst length	1
Send Packet Rate	1/sec
Burst Interval	3 sec
Transmitter Module	
<i>transInterval</i>	5 sec

In consideration of the new transmitter algorithm and the model fault, the redundant data transmission is not needed any longer; thus, both the burst length and the data rate are set as 1. The burst interval is set as 3 seconds to increase the resiliency in case. The *transInterval* collides with other events no more, and thus, can be set an arbitrary value not too short, here 5 seconds should be an appropriate value. Transmitting power

is set values uniformly distributed within two different ranges, by which the link failure probabilities are assumed to be simulated, corresponding to a pre-configured receiving power of 10000 ρ Watt.

4.5.2 Results Evaluation

The comparison of AODV and *ABAF* is analyzed mainly in terms of *message successful delivery ratio, redundancy*.

- **Message Successful Delivery Ratio** The ratio of actually received messages by a destination to sent messages by a source node. Only RREQ is monitored and counted in the simulations for two reasons. One is that the results on RREQ are just right for the performance evaluation of the path discovery process. The other reason is the concern on the model fault and the inappropriate parameter setting.
- **Redundancy** For the same reasons as above, the *transInterval* setting did uncertain interruptions to the data delivery, and consequently produced many unwanted control messages like RRER and ACK; therefore, they were not counted in the redundancy. The total processed messages over the total received messages are considered as the redundancy.

Energy Efficiency should have been practically monitored, if we got more time for this work. It is easy to program a battery module for the purpose; nevertheless, the energy dissipation can still be derived from the redundancy. Each transmitted message consumes a definite transmission power and four receiving power with a probability of $(1 - p)^4$, while TX power and RX power are treated as equal in the scenario. Energy consumption for data computation, sensing and other tasks is neglected, for most of the power at a node is consumed by transceiving. In addition, the energy used for listening to the channel by idle nodes is out of scope of this thesis and also not taken into consideration in this simulation.

Delivery Ratio

Figure 4.7-4.10 show the delivery ratio under various circumstances. The x-axial label, for example *ABAF-k5.sca #1(at line 1)*, means the run #1 of ABAF with a k=5, while the (at line 1) is a numb label produced by the simulator.

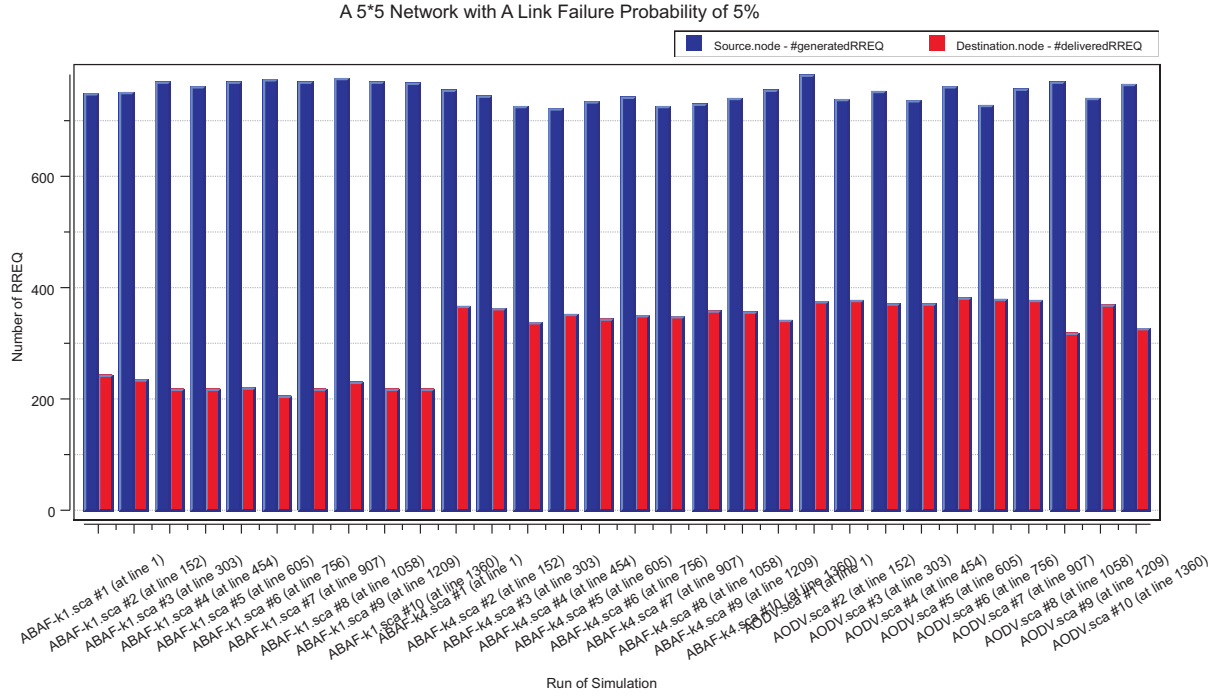


Figure 4.7 Delivery Ratio in A 5×5 Network with A Link Failure Probability of 0.05

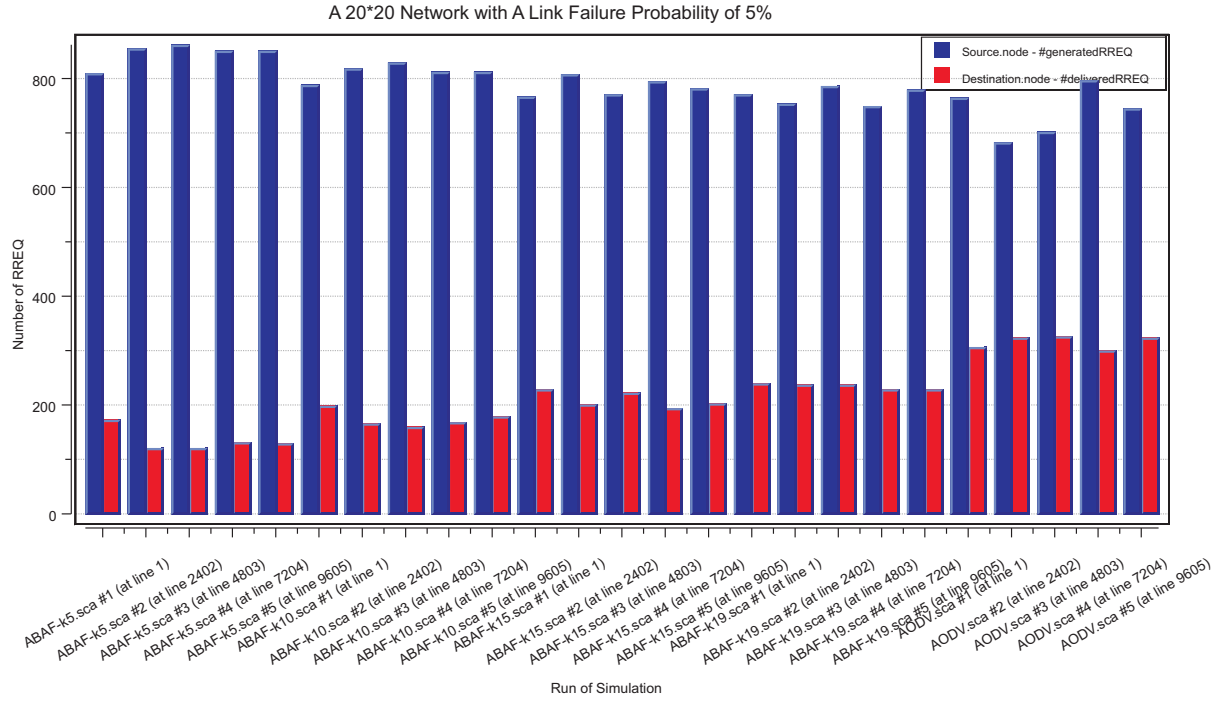


Figure 4.8 Delivery Ratio in A 20×20 Network with A Link Failure Probability of 0.05

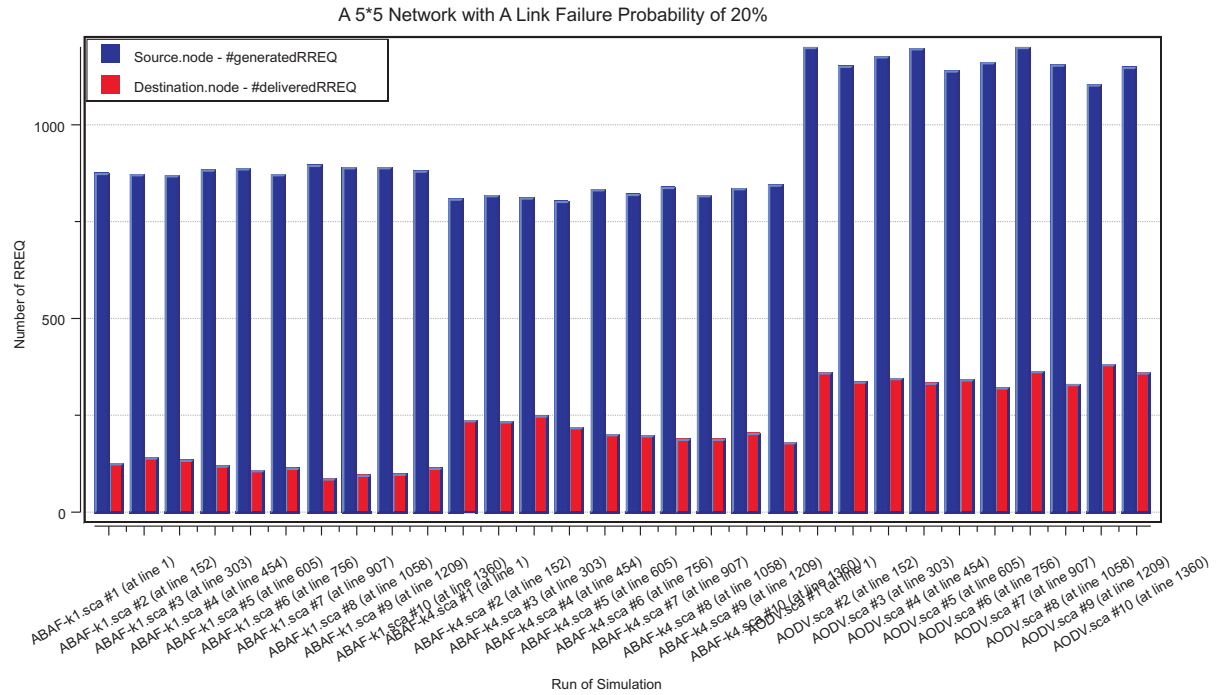


Figure 4.9 Delivery Ratio in A 5×5 Network with A Link Failure Probability of 0.20

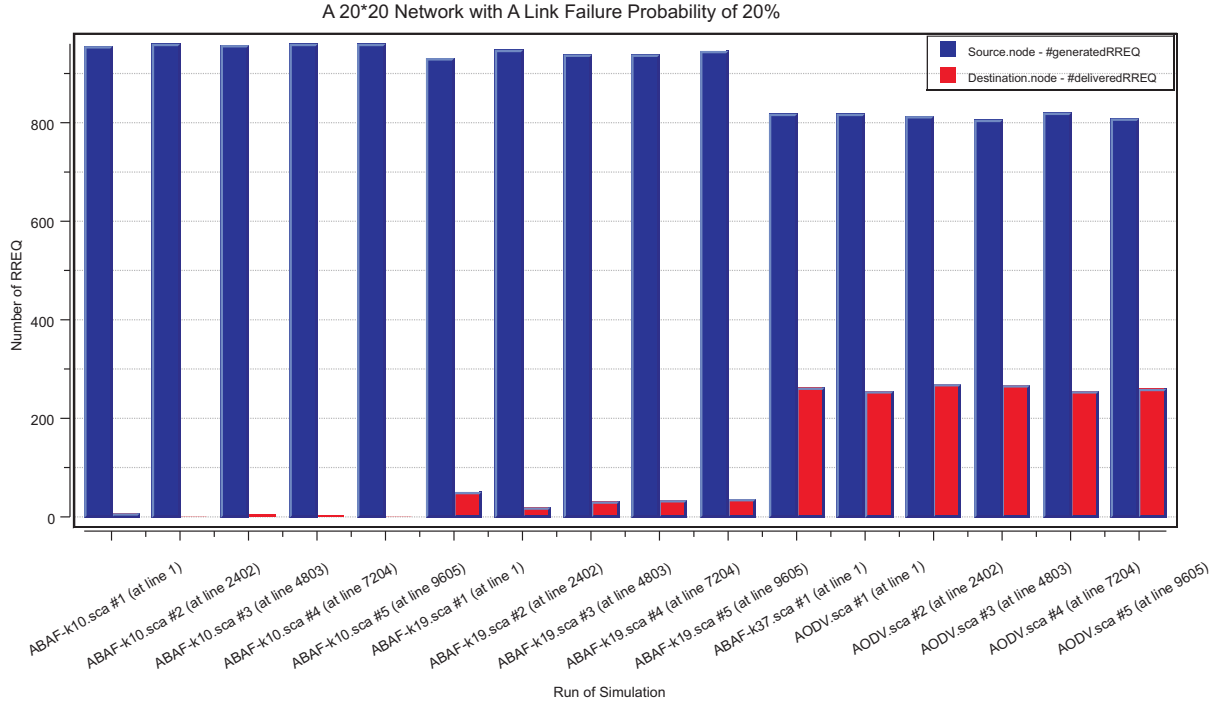


Figure 4.10 Delivery Ratio in A 20×20 Network with A Link Failure Probability of 0.20

The simulations were run for 10 times for each setting in a 5×5 network, and down to 5 times in a 20×20 due to the large simulation redundancy. The figures separately show the generated RREQ by the source nodes over the received RREQ by the destination nodes in each single run. The bar charts obviously reflect the performance decline of the both routing protocols when either the network scale or the link failure increases. As expected, the *ABAF* path discovery algorithm does improve the message delivery ratio by increasing K . In the 20×20 network with a 0.20 link failure (see Figure 4.10), *ABAF* with the K of 37 achieves a same performance as AODV, where *ABAF* equals to a full flooding.

Redundancy

Figure 4.11-4.14 show the redundancy under various circumstances.

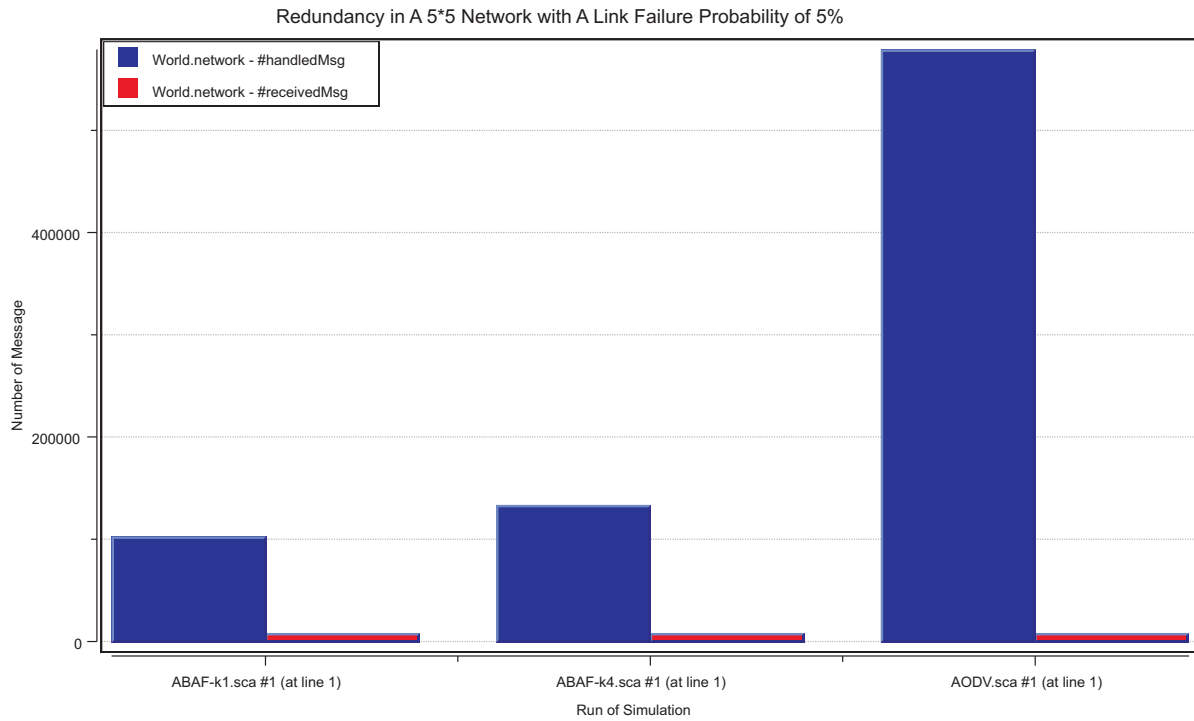


Figure 4.11 Redundancy in A 5×5 Network with A Link Failure Probability of 0.05

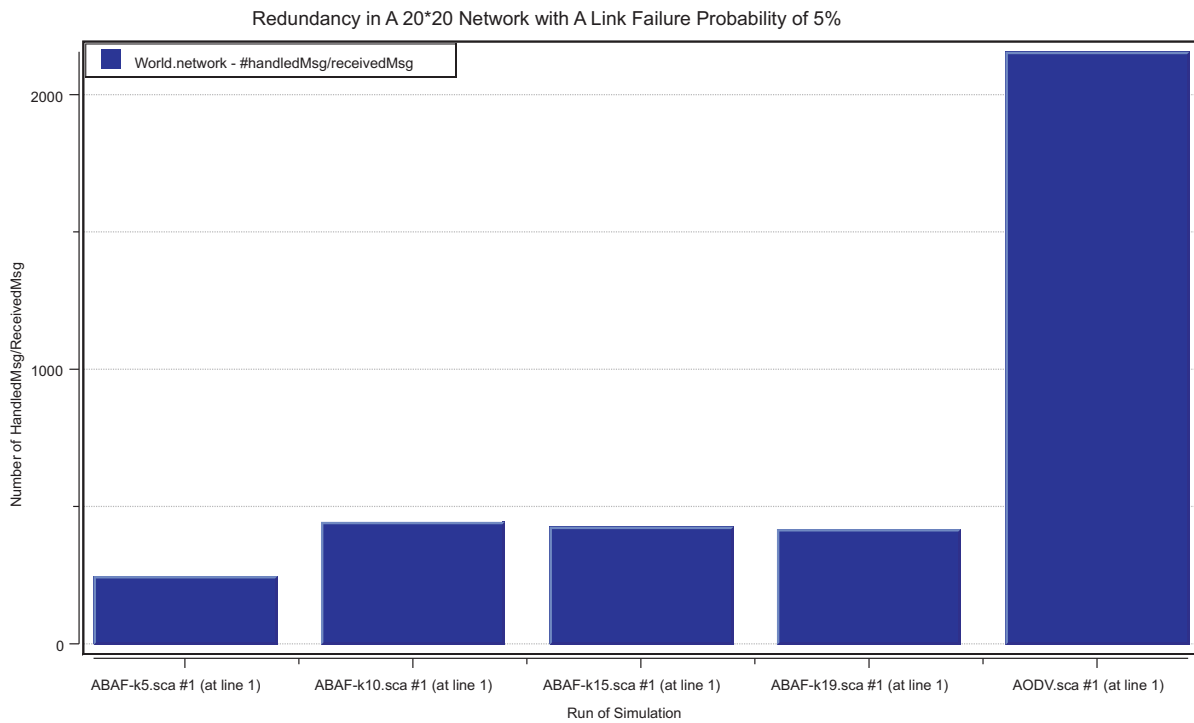


Figure 4.12 Redundancy in A 20×20 Network with A Link Failure Probability of 0.05

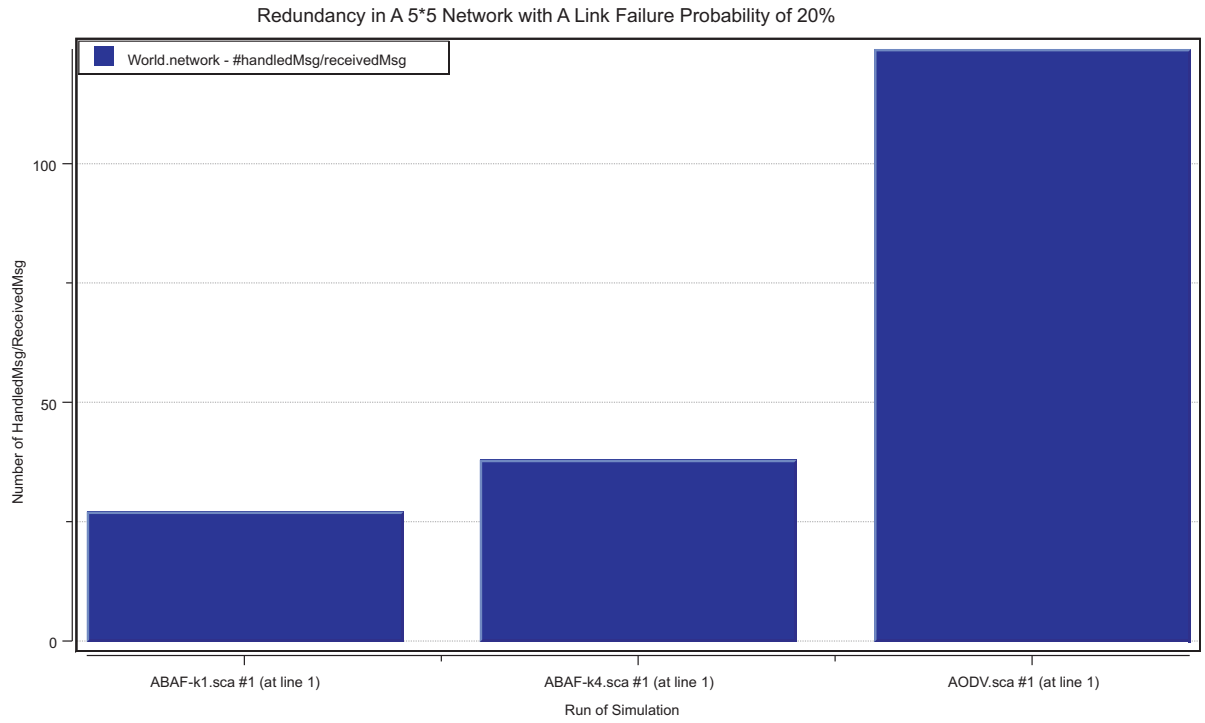


Figure 4.13 Redundancy in A 5×5 Network with A Link Failure Probability of 0.20

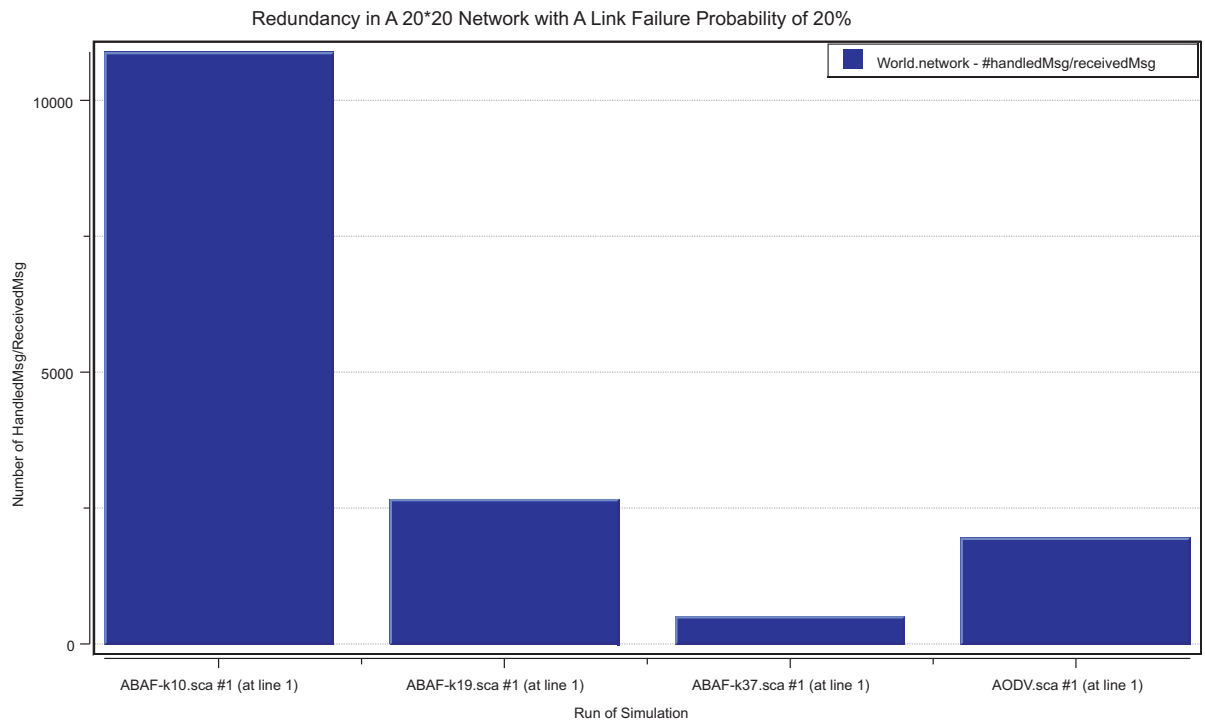


Figure 4.14 Redundancy in A 20×20 Network with A Link Failure Probability of 0.20

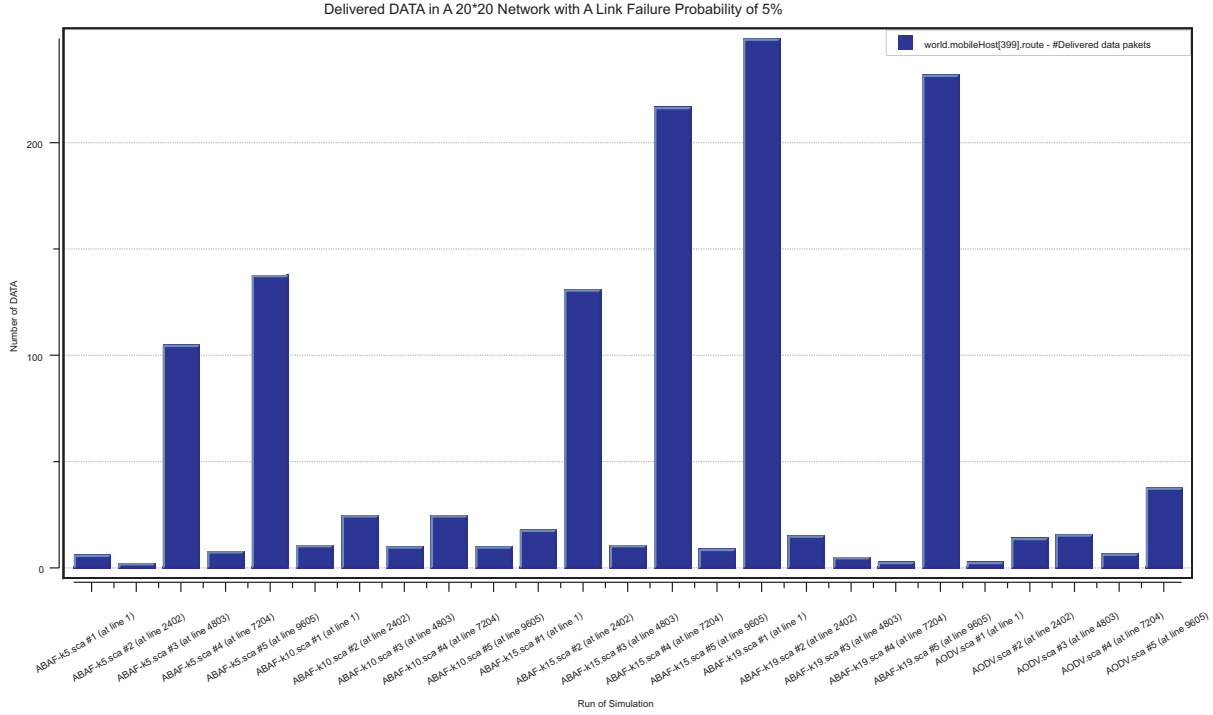


Figure 4.15 Received DATA in A 20×20 Network with A Link Failure Probability of 0.05

Due to the uncertain interferences on the built pathes, the number of the received data packets are very orderless from one simulation to another. As all the charts of the received DATA show irregular results, only an example is shown here in Figure 4.15. However, the results of the total received messages by a destination follow a certain order that reflects the anticipated analysis. Again, because of the interferences, the processed control messages are much more than normal, but still show the expected results. In Figure 4.12-4.14, the number of the received messages is too small, compared to the total handled messages, to be displayed by the simulator tool (only a red line overlapping the bottom line); therefore, the ratio values are put on display instead of the number of respective messages. Figure 4.14 shows a declining trend of redundancy as K increasing in *ABAF* in a large scale network with a highly dynamic environment. The *ABAF* with a k of 37 produces much less redundancy than AODV, while it achieves a similar delivery ratio. The high redundancy in the simulations with a small K of 10 is produced by masses of retransmissions to against the low success rate of path discovery.

Chapter 5

CONCLUSIONS

The attributes of various applications and the dynamic wireless environment make the routing of WSN very challenging. Most of practical application examples and the *near-term* commercial applications list refines the WSN research to a comparatively clearer scope, static manner, within which more resources can be identified and utilized for the design of WSN protocols. Although the application-specific protocol design was advocated at the beginning of the thesis, there still should be several mainstream technologies (standards) that can suffice common attributes of WSN applications; otherwise, it could be much more difficult to popularize WSNs. The protocol proposed in this work is supposed to be such a technology and able to be implemented in the route discovery process of any WSN routing protocols in a static environment.

5.1 SUMMARY OF CONTRIBUTIONS

When improving or designing a technology, it is very important to comprehend the state of arts and define the problems, the goals and the requirements. The literature review in this work covered most of familiar forms of WSN applications and the main taxonomy of the routing protocols. The research on ZigBee specification shows that AODV, designed for MANETs, is still employed as the routing standard due to its simple mechanism and high robustness, despite the high redundancy. A conclusion was obtained afterwards: the protocols are developed more and more complicated while some simple problems in practical applications has not been solved yet.

The addressing scheme proposed by Bhatti and Yue provides a more coherent address structure for WSNs. The new addressing scheme was implemented in the static network model; thereby the *ABAF* routing protocol was developed. The newly proposed routing protocol was clarified step by step and the advantage in the route discovery was proved mathematically. A simulation based on OMNeT++ was carried out afterwards to testify the theoretical results. The analysis of the simulation results leads to a positive conclusion: the path discovery process is adaptive, and a satisfied message delivery ratio can be achieved by setting the flooding counter K a appropriate value; furthermore, the redundancy produced by *ABAF* is obviously much lower than AODV; meanwhile, a low redundancy is supposed to consume less energy. The most important is that the *ABAF* routing protocol is adjustable according to the requirement of the network operation, to achieve a tradeoff between efficiency and robustness, cost and performance. On the other side, the reactive routing protocols are still shown to be less efficient in large scale networks where mesh structures should be suitable; nevertheless, *ABAF* can also be embedded in the routing among parents nodes or cluster heads.

In conclusion, the research in this dissertation contributes to our understanding of the benefits of implementing the systematic address structure in the adaptive hybrid WSN routing strategy. This work really utilized addresses in the WSN routing as a key resource, which makes the WSN network topology systematic and regular; furthermore, a multicast hybrid routing protocol was developed, which can adaptively adjust robustness and efficiency of WSNs according to the requirement of applications and the environment dynamic, and be employed as the discovery method in any static WSNs routing protocols to provide less redundancy while keeping a certain robustness.

5.2 FUTURE WORKS

The *ABAF* routing protocol proposed in this dissertation can be regarded as a draft work, there is still a lot of works to be done. During our implementing the systematic addressing scheme and developing *ABAF*, some specific problems are noticed, and

placed in the future works.

- The single-path success ratio is expressed by the equation $P = (1 - p)^n$ for convenience of analysis; however, it is actually not so practical. In an $(n+1) \times (n+1)$ network, there are n links in each row and column; hence the node $(0, 0)$ has $(2n)!/(n!)^2$ different routes to (n, n) . For a node who has two choices to the next hop, the probability it can forward messages to the next hop successfully is $1 - p^2$. The success ratio of a route thus is as below.

$$(1 - p^2)^{k-a+k} (1 - p)^{a+1} \quad (5.1)$$

Here a is the number of the nodes which have two optional hops towards destination, both X direction and Y direction. Equation 5.1 shall be used to re-evaluate the algorithm from the theoretical point of view in the future work.

- Link failures are not simulated very correctly, as discussed in the previous chapter, due to the model fault and the inaccurate parameter setting. Both of the problems shall be solved in the future work.
- The route maintenance scheme shall be experienced, and the memory usage of the new routing tables and neighbor tables shall be monitored.
- The simulation in this work only simulated the constant traffic from the origin point $(0, 0)$ to (N, N) , but it is not hard to generate traffics among any arbitrary nodes. A finished version of this simulation model should be a group of randomly deployed sensor nodes, which can be self-configured according to the systematic addressing scheme, and start communication based on the *ABAF* routing protocol.
- For the cases that the rectangular section between a source and a destination belongs to a larger network as shown in Figure 3.4, if the partial flooding crosses the diagonal, the following shortest-path routing can make route request messages

access the destination via all four incoming links, as shown in Figure 3.7. This case along with the following special situations will be analyzed in the future works.

- A source and a destination are coaxial;
 - The section between a source and a destination is not a square but a random rectangle.
- Due to the faultiness stated in the chapter of result analysis, the measurement of data packets is not accomplished; therefore, the simulation result is not convincing enough. This aspect of ABAF simulation will be carried out in the future works.
 - In a 2-dimensional grid topology, each node maintains maximum two incoming links; therefore, in the shortest-path stage of *ABAF*, if both of the incoming links are disconnected, a node will become inaccessible unless flooding. To increase the robustness under this circumstance, the solution of dead end problem proposed in [21] and 2-hop neighbors in [20] can be considered to add on in the future works.
 - A closed-loop control system could be embedded into the routing process to adjust the flooding counter K automatically according to the wireless channel condition.
 - The addressing scheme has some issues as discussed in the chapter 2, which could be a part of the future works or another Master's thesis worthy work.

Being a hot topic in the field of wireless networks, WSNs attract much attention. Although masses of research has been done, there is still a distance to standardized commercial applications. Due to the properties of applications, protocols and architectures of WSNs require robustness and efficiency more than other networks; therefore, the approach of cross-layer design has obvious advantages; in addition, implementation of currently proposed protocols in practice is considered as another direction of the WSN research in the future.

REFERENCES

- [1] N. Baker, “Zigbee and Bluetooth Strengths and Weaknesses for Industrial applications,” *IEE Computing & Control Engineering*, pp. 20–25, April/May 2005.
- [2] H. Karl and A. Willig, *Protocols and Architectures for Wireless Sensor Networks*. John Wiley & Sons, Ltd, 2005.
- [3] *ZigBee Specification*. ZigBee Alliance, Inc., r17 ed., January 2008.
- [4] B. Heile, “Zigbee Alliance Tutorial,” ZigBee Alliance, <http://www.zigbee.org>, September-November 2005.
- [5] K. Sohraby, D. Minoli, and T. Znati, *WIRELESS SENSOR NETWORKS Technology, Protocols, and Applications*. John Wiley & Sons, Inc., 2007.
- [6] J. M. Rabaey, M. J. Ammer, J. L. da Silva, D. Patel, and S. Roundy, “Picoradio Supports Ad Hoc Ultra-Low Power Wireless Networking,” *IEEE Computer*, vol. 33, no. 7, pp. 42–48, 2000.
- [7] M. Holler, “Camalie Net Wireless Sensing.” <http://camalienetworks.com/>.
- [8] N. Bulusu and S. Jha, eds., *WIRELESS SENSOR NETWORKS A Systems Perspective*. ARTECH HOUSE, INC., 2005.
- [9] K. A. D. et al., “Environmental Studies with the Sensor Web: Principles and Practice,” *Sensors*, vol. 5, pp. 103–117, 2005.
- [10] J. L. Hill, *System Achitecture for Wireless Sensor Networks*. PhD thesis, Department of Computer Science, University of California, Berkeley, 2003.

- [11] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," in *Proceedings of the 33rd Hawaii International Conference on System Sciences*, 2000.
- [12] E. M. Royer and C.-K. Toh, "A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks," *IEEE Personal Communications*, pp. 46–55, April 1999.
- [13] J. N. Al-Karaki and A. E. Kamal, "Routing Technologies in Wireless Sensor Networks: A Survey," *IEEE Wireless Communications*, vol. 11, pp. 6–28, December 2004.
- [14] C. Schurgers and M. B. Srivastava, "Energy Efficient Routing in Wireless Sensor Networks," in *Proceedings of the IEEE Military Communications Conference (MilCom'01): Communications for Network-Centric Operations-Creating the Information Force*, (McLean, VA), October 2001.
- [15] S. Tilak, N. B. Abu-Ghazaleh, and W. Heinzelman, "A Taxonomy of Wireless Micro-Sensor Network Models," *Mobile Computing and Communications Review*, vol. 1, pp. 28–36, April 2002.
- [16] K. Akkaya and M. Younis, "A survey on routing protocols for wireless sensor networks," *Ad Hoc Networks*, vol. 3, pp. 325–349, 2005.
- [17] W. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," in *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99)*, (Seattle, WA), August 1999.
- [18] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," in *Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'00)*, (Boston, MA), August 2000.

- [19] G. Bhatti and G. Yue, “A Structured Addressing Scheme for Wireless Multi-Hop Networks,” Tech. Rep. TR2005-149, Mitsubishi Electric Research Laboratories, June 2006.
- [20] I. Stojmenovic and X. Lin, “Loop-Free Hybrid Single-Path/Flooding Routing Algorithm with Guaranteed Delivery for Wireless Networks,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, pp. 1023–1032, October 2001.
- [21] L. Zou, M. Lu, and Z. Xiong, “A Distributed Algorithm for the Dead End Problem of Location Based Routing in Sensor Networks,” *IEEE Transactions on Vehicular Technology*, vol. 54, pp. 1509–1522, July 2005.
- [22] D. B. et al, “Flat vs. Hierarchical Network Control Architecture,” in *ARO/DARPA Workshop on Mobile Ad-Hoc Networking*, 1997.
- [23] “Internetworking Technology Handbook,” Cisco Systems, Inc., <http://www.cisco.com>.
- [24] “Zigbee Architecture Overview,” ZigBee Alliance, <http://www.zigbee.org>, September 2005.
- [25] S. Nesargi and R. Prakash, “MANETconf: Configuration of Hosts in a Mobile Ad Hoc Network,” in *Proceedings of IEEE Infocom*, 2002.
- [26] C. E. Perkins, J. T. Malinen, R. Wakikawa, E. M. Belding-Royer, and Y. Sun, “IP Address Autoconfiguration for Ad Hoc Networks,” in *draft-ietfmanet-autoconf-01.txt*, IETF Internet Draft, November 2001.
- [27] M. Mohsin and R. Prakash., “IP Address Assignment in a Mobile Ad Hoc Network,” in *Proceedings of the IEEE Military Communications Conference (Mil-Com’02)*, 2002.
- [28] A. Varga, *OMNeT++ User Manual: Discrete Event Simulation System*. <http://www.omnetpp.org>, version 3.0 ed., December 2004.

- [29] A. Kroller, D. Pfisterer, C. Buschmann, S. P. Fekete, and S. Fischer, “Shawn: A new approach to simulating wireless sensor networks.” <http://arxiv.org/abs/cs/0502003>, February 2005.
- [30] N. Concer, *Ad hoc Sim*. <http://www.cs.unibo.it/~concer/>, 1.1 ed., January 2005.
- [31] D. Yu and H. Li, “A Model for Performance Analysis of Mobile Ad-Hoc Networks,” in *Proceedings of International Conference on Telecommunications*, (Peking), June 2002.
- [32] H. G. Goh, M. L. Sim, , and H. T. Ewe, “PERFORMANCE STUDY OF TREE-BASED ROUTING ALGORITHM FOR 2D GRID WIRELESS SENSOR NETWORKS,” in *Proceedings of the 12th IEEE International Conference on Networks*, 2004.
- [33] Y. M. Lu and V. W. S. Wong, “An energy-efficient multipath routing protocol for wireless sensor networks,” *INTERNATIONAL JOURNAL OF COMMUNICATION SYSTEMS*, vol. 20, pp. 747–766, 2007.
- [34] W. Li, H. Sirisena, and K. Pawlikowski, “An Address-Based Routing Scheme for Static Applications of Wireless Sensor Networks,” in *Proceedings of Australasian Telecommunication Networks and Applications Conference (ATNAC’07)*, (Christchurch), December 2007.